هدف این بخش از پروژه آشنایی و کسب تجربه شما در زمینه تحلیل، استخراج ویژگی‌ها و رده‌بندی می‌باشد. لطفا زودتر کل توضیح پروژه را مطالعه کرده و هر گونه ابهام را از اساتید حل تمرین یا استاد درس سوال بفرمایید.

ساختار پوشه‌ها و فایل‌های مورد نیاز:

- data: داده‌های جمع‌آوری شده و تمیز شده از مرحله اول پروژه بعلاوه پردازش‌های جدید این مرحله.
- src: تمام کدهای نوشته شده برای پروژه بدون استثناء.
- reports: تمام گزارش‌های تولید شده برای پروژه بصورت اتوماتیک مثل نمودار، اطلاعات جدول، نمونه جمله،... این گزارش‌ها برای استفاده بصورت مستقیم و بدون تغییر در فایل گزارش نهایی در قالب لاتک استفاده می‌شوند که با دستورهای لازم مثل includegraphics، import، input از همین پوشه reports در فایل لاتک جا داده می‌شوند. بدون کپی کردن دستی.
- models: کلیه مدل‌ها آموزش داده شده.
- <experiment_name>: پوشه کاری (working directory) برای آزمایش/بخش خواسته شده که شامل فایل‌های میانی تولید شده می‌باشد. نام این پوشه برای هر بخش در ابتدای پاراگراف به لاتین آمده است.
- run.py/bat/sh: یک فایل تنها که با اجرای آن (در صورت پاک کردن تمام پوشه‌ها بجز data و src و latex) کلیه کدهای لازم اجرا شده و گزارش‌ها و مدل‌های لازم تولید شد و فایل گزارش نهایی مجددا تولید شود.
- run.log: این فایل در ریشه ریپازیتوری بوده و logهای سطح اول مربوط به run در رابطه با صدا زدن کد برای اجرای بخش‌های مختلف و بررسی اتمام موفقیت‌آمیز هر بخش و تولید خروجی‌های لازم آن بخش در این فایل گزارش شود.
- logs: تمام کدهای شما باید در این پوشه جزئیات کافی را log کنند بطوریکه در صورت متوقف شدن کد یا پیش‌آمد خطا بتوان از این پوشه خطایابی شود. در این پوشه به ازای هر دستور/task/کار جداگانه لازم است فایل log جداگانه با اسم متناسب موجود باشد.
- latex: متن گزارش شما به فارسی یا انگلیسی. دقت کنید نمودارها و جدول‌ها تولید شده توسط کد شما باید مستقیما از پوشه reports ارجاع داده شده و جای‌سازی بشوند و داخل اینجا کپی نشوند.
- report_final.pdf: گزارش کامپایل شده نهایی.

۱. **بخش word2vec:** با استفاده از کد Word2Vec مربوط به تمرین A2 بردار کلمات را برای هر کدام از دسته‌های داده بصورت جداگانه آموزش دهید و مدل خروجی را در پوشه models و با نام <label>.word2vec.<fileformat_extension> ذخیره کنید. مثلا conservative_news.word2vec.npy. با اجرای اسکریپت/کد اصلی شما باید فایل‌های مدل بصورت خودکار و با نام درست در پوشه مورد نظر ذخیره شوند. بدون هیچگونه کار دستی. همچنین کد مورد نیاز برای بارگذاری/load مدل و query از آن برای تولید نمودار یا گزارش‌های این بخش باید در پوشه src موجود بوده و نتایج مورد استفاده در گزارش در پوشه‌ای به نام reports با فرمت لازم (csv, png, txt...) بصورت خودکار ذخیره شود.

- بردارهای کلمات مشترک بین دسته‌ها را با هم مقایسه و تحلیل کنید. از کلمات مشترک بین دسته‌ها، کدامیک بردار مشابهی در هر دو دسته دارند و کدامیک متفاوت است. علت تشابه یا تفاوت چیست. بایاس را در بردارها بررسی کنید. با ذکر مثال و نمودار/جدول نتیجه تحلیل را در مستند این بخش گزارش کنید. روش مقایسه/تحلیل بر عهده شماست. مثلا مقایسه شباهت کسینوسی، نزدیک‌ترین همسایه‌ها، ... .
- همچنین برای استفاده در مراحل بعد یک مدل word2vec روی تمام داده‌ها با هم آموزش داده و به نام all.word2vec.npy ذخیره کنید.

۲. **بخش tokenization:** مانند تمرین A4 از کتابخانه SentencePiece برای آموزش Tokenize کردن داده با حداقل ۴ اندازه متفاوت (از خیلی کم تا خیلی زیاد نسبت به اندازه داده شما) روی داده خام تمیز شده اجرا و ارزیابی کنید. داده خود را به ۵ بخش تقسیم کرده

و در هر مرحله ۵ بار آموزش و ارزیابی کنید. در هر مرتبه روی ۴ قسمت از ۵ قسمت آموزش داده و درصد توکن‌های Unk را محاسبه کرده و هر کدام از درصدها بعلاوه متوسط آنها را در یک جدول به تفکیک «تعداد توکن ورودی برای آموزش SentencePiece» گزارش کنید. همچنین در هر یک از موارد توکن‌های ایجاد شده را بررسی کرده و با ذکر مثال نتیجه Tokenize کردن با مقادیر مختلف را تحلیل کرده و نهایتا یک اندازه را برای Tokenizer انتخاب کنید. مانند بخش‌های قبل کد استفاده شده برای اجرا و تحلیل این آزمایش‌ها همگی باید در پوشه src موجود بوده و بگونه‌ای نوشته شده باشد که همه آزمایش‌های یکی-پس-از-دیگری بتوانند اجرای مجدد شده و نتایج گزارش در پوشه reports با نام مناسب تولید شود. پس از انتخاب بهترین تنظیم Tokenizer لازم است مدل نهایی بصورت خودکار (تنظیمات SentencePiece را hard-code کرده) به پوشه model کپی شود.

۳. **بخش parsing**: در این بخش لازم است با استفاده از کد تمرین شماره ۳ تمام داده را تحلیل زبانی بکنید. داده لازم به زبان مورد نظر را از https://universaldependencies.org دانلود کرده مدل dependency_parser را روی آن آموزش داده و سپس روی داده خود اعمال کنید. تعداد ۱۰ جمله (یا بیشتر) را بصورت دستی dependency_parse آنرا مشخص کرده و روی این ۱۰ جمله دقت parser را محاسبه و گزارش کنید. چند جمله را به عنوان نمونه در گزارش خود آورده و در صورت داشتن خطا علت آن را تحلیل کنید.

۴. **بخش language_model**: برای این بخش می‌توانید کد تمرین شماره ۴ را تغییر داده یا کد آماده کوتاهی در pytorch استفاده کنید. برای هر کدام از دسته‌های داده خود یک مدل زبانی جداگانه به نام <label>.language_model آموزش داده و تعدادی جمله به ازای هر کدام از دسته‌ها تولید کنید. آیا تفاوت جمله‌های تولید شده با انتظار شما تطابق دارد. لطفا نمونه‌ها را در گزارش خود آورده و تحلیل کنید.

۵. **بخش fine_tuning**: مدل BERT یا ParsBERT را روی هر کدام از دسته‌های داده خود بصورت جداگانه fine_tune کرده و در قالب مدلی به نام <label>.bert_lm ذخیره کنید. سپس از این مدل‌ها برای تولید جملات استفاده کنید. آیا جملات تولید شده تفاوت مورد انتظار شما را دارند؟ جملات را در گزارش خود ارائه و تحلیل کنید. همچنین مدل BERT یا ParsBERT را برای رده‌بندی داده‌ها روی کل داده‌ها fine-tune کنید و به نام bert_classification_lm ذخیره کنید.

۶. **بخش feature_engineering**: برای این بخش (و بخش‌های آینده) داده خود را به سه دسته train/dev/test با درصدهای 80/10/10 بصورت متوازن برای هر برچسب تقسیم کنید. سپس دو معماری ساده برای رده‌بندی/classification داده‌ها در نظر بگیرید. یک معماری که تمام فیچرهای جمله را یکجا دریافت کرده و رده‌بندی کند. یک مدل دیگر که فیچرها را یکی-یکی دریافت می‌کند. بستگی به نوع فیچر از هر کدام از معماری‌ها که لازم است استفاده کرده و با فیچرهای زیر بصورت جداگانه آموزش داده و نتیجه را برای داده train/validation/test در epochهای مختلف در یک نمودار گزارش کنید. چنانچه فقط یک معماری در نظر بگیرید که برای همه فیچرها قابل استفاده باشد، اشکالی ندارد. تمام فیچرهای زیر بصورت جداگانه روی همین یک نمودار رسم شود. همچنین نتایج در یک جدول به تفکیک train/test/validation و feature گزارش شود.

- sentence_length: طول جمله (یا واحد مناسب برای رده‌بندی) را به عنوان تنها فیچر در نظر بگیرید.
- word_length: مجموعه طول کلمات به ترتیب.
- parse_only: از فیچرهای مربوط به dependency_parse تعدادی فیچر مثل ارتفاع درخت، حداکثر تعداد بچه‌ها، ... (بدون استفاده از هر گونه کلمه/محتوا فقط فیچرهای ساختاری) استخراج کرده، با هم concat کرده و به عنوان فیچر استفاده کنید.
- words: هر کلمه را به یک عدد منحصر به فرد تخصیص داده و به عنوان فیچر استفاده کنید.
- lexicalized_parse: یک فیچر قابل استخراج از درخت مثل کلمه و ارتفاع، یا کلمه و فرزند... به اختیار خود انتخاب کرده کلمه را با عدد آن جایگزین کرده، با هم concat کرده و به عنوان فیچر استفاده کنید.
- word bi-grams: عدد هر کلمه و کلمه قبل را با هم concat کرده و به عنوان یک فیچر استفاده کنید.

- word2vec: از بردارهای word2vec به عنوان فیچر متناظر با هر کلمه استفاده کنید.
- word2vec_bigram: بردار هر کلمه را با کلمه قبلش concat کرده و به عنوان یک فیچر استفاده کنید.
- BERT/ParsBERT: از بردارهای BERT یا ParsBERT به تنهایی به عنوان فیچر استفاده کنید.
- FineTuned_BERT: از بردارهای FineTune شده BERT یا ParsBert به عنوان فیچر استفاده کنید.

۷. **بخش model_architecture**: حال که فیچرهای مختلف را امتحان کرده و میزان موفقیت آنها در رده‌بندی را امتحان کردید، نوبت به انتخاب معماری مناسب می‌باشد. تعدادی از فیچرهای بخش قبل را به عنوان فیچر به شکل دلخواه/مختلف با هم ترکیب کرده و حداقل ۳ معماری مختلف که حداقل یکی از آنها مبتنی بر Transformer باشد را برای رده‌بندی داده‌ها با هم آموزش، آزمون و مقایسه کنید. مانند قسمت‌های قبل مدل‌های نهایی در پوشه models و گزارش‌های در پوشه reports ذخیره شوند. نتایج را در گزارش خود ارائه و تحلیل کنید.

۸. **بخش interpretation**: این بخش اختیار می‌باشد و دارای حداکثر یک نمره مثبت از ۲۰ نمره نهایی می‌باشد. یکی از معماری‌های بالا را انتخاب کرده و آنرا تحلیل و تفسیر کنید از جهت اینکه مدل چه چیزی یاد گرفته است. برای این مورد در مورد interpretability می‌توانید جستجو کرده و از ابزارها و منابع آماده استفاده کنید. چنانچه از attention استفاده می‌کنید استخراج وزن‌های attention می‌تواند نقطه شروع برای این کار باشد.

در اکثر موارد استفاده از قطعه کدهای آماده آنلاین به شرط آشنایی کامل شما با کد و ذکر منبع بلامانع می‌باشد. چنانچه در موردی شک دارید سوال کنید.

موفق باشید
اعتمادی