هدف این بخش از پروژه آشنایی و کسب تجربه شما در زمینه تحلیل، استخراج ویژگی‌ها و رده‌بندی می‌باشد. خروجی این فاز در همان ریپازیتوری مرحله اول تحویل گرفته می‌شود. خروجی مرحله اول (Phase1_Report.pdf) باید مثل قبل قابل تولید باشد و همان مستند تحویل داده شده در گریداسکوپ را تولید کند. تنها تفاوت قابل قبول تغییر داده یا اصلاح برنامه‌های محاسبه می‌باشد که تفاوت آن فقط در جداول و/یا نمودارهای گزارش خواهد بود. هر گونه تغییر یا اصلاح دیگر در گزارش باید در خروجی مرحله دوم (Phase2_Report.pdf) آورده شود. ساختار پوشه‌ها علاوه بر ساختار/پوشه‌های فاز یک، شامل بخش‌های زیر نیز می‌باشد.

- `experitments/<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_phase2.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 فاز ۲ 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.
- `models`: لازم است مدل‌های آموزش داده شده در این پوشه با نام‌های منحصر به فرد ذخیره شوند.
- `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.
- `Phase2_Report.pdf`: گزارش کامپایل شده نهایی.

۱. **بخش word2vec (۱۰)**: 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 (۵)**: 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 language_model (۲۰)**: برای هر کدام از دسته‌های داده خود یک مدل زبانی (شما می‌توانید هر مدل اتوریگرسیو زبانی را برای این بخش انتخاب کنید) انتخاب کرده و روی داده خود تنظیم دقیق (finetune) کرده و جداگانه به نام <label>.language_model در پوشه models ذخیره کنید. سپس تعدادی جمله به ازای هر کدام از دسته‌ها تولید کرده و در پوشه stats ذخیره کرده و در گزارش خود آورده و تحلیل کنید. آیا تفاوت جمله‌های تولید شده با انتظار شما تطابق دارد؟ برای آموزش مدل‌های زبانی می‌توانید از این نوتبوک موجود در کگل کمک بگیرید. همچنین می‌توانید از کتابخانه simpletransformers نیز استفاده کنید.

۴. **بخش feature_engineering (۱۵)**: 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.
- words: هر کلمه را به یک عدد منحصر به فرد تخصیص داده 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.

۵. **بخش model_architecture (۲۵)**: 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.

۶. **بخش data augmentation (۱۵)**: در این بخش لازم است که در سایت Open AI برای دسترسی به Chat-GPT ثبت‌نام کنید و با استفاده از API آن به افزودن داده خود بپردازید. پس از اتمام تولید داده‌ها، Promptهای استفاده شده برای هر دسته از داده را همراه با نمونه‌هایی از داده اصلی و داده‌های تولید شده توسط Chat-GPT و تحلیل آن‌ها را در گزارش خود قرار دهید. لازم به ذکر است این بخش در صورتی نمره کامل دارد که داده اضافه شده دارای کیفیت قابل قبولی باشد.

۷. **انجام رده بندی توسط OpenAI (۱۰)**: در این بخش لازم است Promptی طراحی کنید که رده‌بندی را بصورت Zero-Shot و/یا Few-Shot توسط OpenAI انجام دهد و دقت آن به عنوان OpenAI Baseline با معماری‌های بالا مقایسه شود.

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