به نام خدا

دوره کارآموزی هوش مصنوعی

پردازش زبان طبیعی

بخش دوم پروژه – طبقه‌بندی با روش‌های کلاسیک

محدثه رهنما – زینب تقوی

آبان الی دی ۹۹

تغییرات در پیش‌پردازش

به منظور کاهش حداقلی تعداد کلمات متمایز در مجموعه داده‌ی به دست آمده، توکن‌های یک و دو حرفی حذف شدند. از طرفی با بررسی مجدد داده‌ها به دست آمده، مشخص شد که برخی از اسناد از حد معمول طولانی‌تر هستند. در چنین مواردی اصل مطالب بیشتر در دو بخش پرداخته می‌شود: ابتدای متن که توضیح کلی درباره‌ی موضوع مورد بحث بیان می‌شود و انتهای متن که با جمع‌بندی مطالب خاتمه می‌یابد. با در نظر گرفتن این مهم، تنها توکن‌های ابتدا و انتهای متن مورد پردازش قرار می‌گیرد. در این داده‌ها میانگین طول اسناد حدود ۳۰۰ کلمه است و بنابراین در اسناد طولانی، حدود ۱۵۰ کلمه از ابتدا و ۱۵۰ در انتهای متون انتخاب شده و برای پردازش در ادامه‌ی مراحل مورد استفاده قرار می‌گیرد. لازم به ذکر است که نتایج به دست آمده در این حالت با نتایج حاصل از پردازش کل متون تفاوت چندانی نداشته است و بنابراین به نظر می‌رسد حذف بخش‌های میانی در مقالات طولانی راه حل مناسبی در کاهش تعداد کلمات متمایز و مفید است. واضح است که حداقل کردن تعداد کلمات در بهبود زمان و مکان (حافظه) مؤثر است.

استخراج ویژگی

استخراج ویژگی با محاسبه‌ی tf-idf کلمات صورت گرفته است. در این روش هر سند با یک بردار ویژگی نشان داده می‌شود که ابعاد آن با تعداد کل کلمات متمایز در مجموعه داده برابر است و مقدار هر ویژگی برابر tf-idf هر کلمه در آن سند است. استخراج ویژگی با کتاب‌خانه sklearn و تابع tfidfVectorizer() انجام شد. در این مرحله نیز با هدف این که ابعاد ویژگی‌ها کاهش یابد، tf-idf کلمات پرکاربرد به دست می‌آید. طبق قانون *Zipf's* (Li, Sheng, Luan, & Chen, 2009) ۲۰ درصد کلمات کاربرد بیشتری دارند. بدین سبب پارامتر max_features در این تابع پارامتری برابر با ۲۰٪ تعداد کلمات متمایز قرار داده شده است. در تابع tfidfVectorizer() اگر پارامتر max_features مقداردهی شود، کلماتی که tf بالاتری دارند در نظر گرفته می‌شود. به منظور کاهش بُعد ویژگی‌ها نیز از تابع truncatedSVD() استفاده شده است.

طبقه‌بندی

الگوریتم‌هایی که برای دسته‌بندی انتخاب شده‌اند عبارتند از ماشین بردار پشتیبان، knn ، جنگل تصادفی و نایو بیزین. در الگوریتم ماشین بردار پشتیبان تابع هسته RBF انتخاب شده و پارامترها با مقادیر پیش فرض انجام شده است. در الگوریتم knn تعداد همسایه‌ها عدد ۵ مناسب بوده است و در نایو بیزین توزیع گوسین انتخاب شده است. برای پیاده سازی الگوریتم‌های یاد شده از کتاب‌خانه‌ی sklearn استفاده شده است.

نتایج

در این بخش یافته‌های حاصل از الگوریتم‌های طبقه‌بندی ارائه می‌شود و تأثیر پردازش‌های مختلف در کاهش ابعاد و متعادل کردن نمونه‌ها بررسی می‌شود.

در جدول زیر نتایج به دست آمده با در نظر گرفتن تمامی کلمات -از جمله اسناد طولانی- خلاصه شده است.

الگوریتم	دقت	F1-score
SVM	٪۲۵	٪۲۸
Naïve Bayesian	٪۱۷	٪۱۸
Random forest	٪۲	٪۲۶
KNN	٪۲۷	٪۳۲

پس از بررسی طول کلمات هر سند، مشخص شد که میانگین طول کلمات برابر با ۳۲۳/۹۷ است. بر همین اساس در اسنادی که تعداد توکن‌های آن‌ها بیش از این تعداد باشد، حدود ۱۵۰کلمه از ابتدا و انتهای سند انتخاب می‌شود و نتایج حاصل در جدول زیر آمده است.

الگوریتم	دقت	F1-score
SVM	٪۳۱	٪۳۶
Naïve Bayesian	٪۳	٪۲۷
Random forest	٪۱۵	٪۱۶
KNN	٪۲۵	٪۳۱

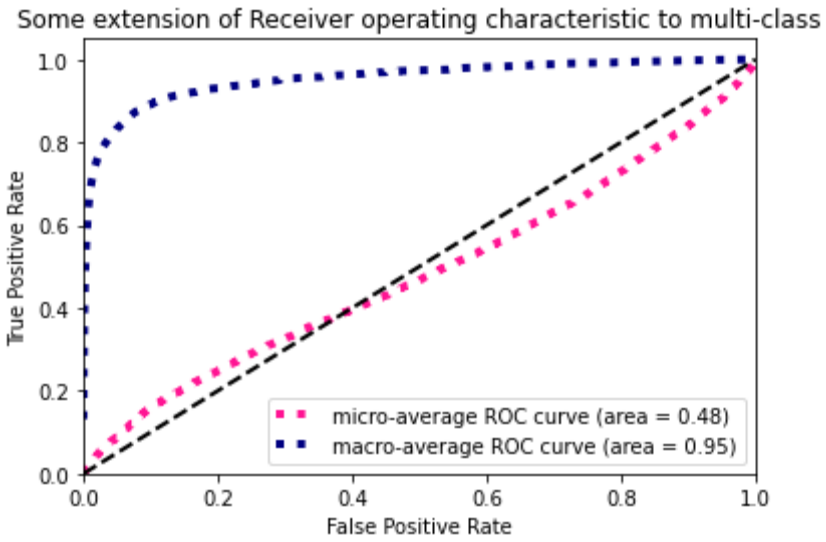
بدیهی است برای برخی از برچسب‌های موردنظر نمونه‌های کمتری به دست آمده است و همین سبب عدم تعادل (imbalanced data) می‌شود. یکی از راه‌حل‌های مشکل مذکور، افزایش نمونه‌های کلاس‌هایی که تعداد نمونه‌های آن‌ها نسبت به سایر کلاس‌ها کمتر است. تولید نمونه‌های مشابه در مسائلی ممکن است که نمونه‌ها کمی باشند. در مقابل یافتن نمونه‌های نزدیک به نمونه‌های هر کلاس در این مسئله به معنای یافتن کلمات مشابه و هم معنی است و به سادگی مسائل کمی نیست. راه حل دیگر می‌تواند مربوط به کلاس‌هایی باشند که نسبت به سایر کلاس‌ها نمونه‌های بیشتری دارند و ممکن است کاهش نمونه‌ها در کلاس‌های یادشده سبب تعادل در داده‌ها و بهبود عملکرد الگوریتم‌ها شود. گرچه نتایج حاصل از این آزمایش نشان می‌دهد که کاهش نمونه‌ها اثر مناسبی نداشته است. جدول زیر یافته‌های به دست آمده از این آزمایش است.

الگوریتم	دقت	F1-score
SVM	٪۱۷	٪۲۰
Naïve Bayesian	٪۵	٪۲۸
Random forest	٪۹	٪۱۲
KNN	٪۱۹	٪۲۵

بهترین نتایج در حالتی به دست آمد که تمامی اسناد به طول میانگین حدود ۳۰۰ کلمه پردازش شده و بهترین مدل با الگوریتم ماشین بردار پشتیبان حاصل شد. مقادیر معیارهای ارزیابی در این حالت به شرح زیر است:

میانگین	صحت	بازخوانی	F1-score
Micro	٪۷۸	٪۲۷	٪۴۱
Macro	٪۶۴	٪۲۱	٪۲۸
Weighted	٪۷۴	٪۲۷	٪۳۶

میانگین دقت به دست آمده برابر با ٪۳۰ بوده است و نمودار ROC نیز به شکل زیر است:



به دلیل طولانی شدن مطالب، نتایج به دست آمده و نمودارهای مربوط به هر کلاس در این سند ارائه نشده و در کد step_by_step3 در گوگل درایو مشترک قابل مشاهده است.

چالش ها

ذخیره‌ی ویژگی‌ها

پس از محاسبه‌ی مقادیر tf-idf با تابع tfidfVectorizer()، حاصل ویژگی‌ها در قالب ماتریس‌های spase هستند. در صورتی که قصد ذخیره‌سازی این ویژگی‌ها به فرمت‌هایی چون csv داشته باشیم، می بایست ویژگی‌ها به آرایه تبدیل شده (با کتابخانه‌ی numpy) که به دلیل تعداد بالای نمونه‌ها و ابعاد ویژگی‌ها، این آرایه اندازه‌ی بسیار بالایی دارد که سبب crash کردن حافظه‌ی ram در فضای google colab می‌شود. همچنین این مشکل در کاهش ابعاد به روش‌هایی چون pca نیز وجود دارد.

یکی از راه‌حل‌های برطرف کردن این مشکل، کاهش بعد با روش svd است. تابع truncatedSVD() می‌تواند ورودی در قالب ماتریس sparse دریافت کند و پس کاهش ابعاد ویژگی‌ها می‌توان آن را به آرایه تبدیل کرد و در قالب دلخواه مانند csv ذخیره نمود. ذخیره‌ی ویژگی‌ها از اتلاف وقت جلوگیری کرده و نیازی به اجرای مجدد این مرحله نیست.

کاهش بعد

یکی از نکات مهم در کاهش بعد ویژگی‌ها، تعیین تعداد ابعاد جدید است. در تابع truncatedSVD() برای یافتن تعداد مناسب کامپوننت‌ها، این است که مقادیر متفاوت را امتحان کنیم و مقدار svd.explained_variance_ratio_.sum() را به دست آوریم. اگر این پراکندگی بیشتر از ۰/۹ یا ۰/۹۵ باشد یعنی تعداد کامپوننت‌ها نسبتاً درست مقداردهی شده و ابعاد واقعی داده‌ها برابر با تعداد تعیین شده است. در این مسئله مقادیر مختلف از ۱۰۰ بعد تا ۱۰۰۰ بعد امتحان شد و عدد ۷۰۰۰ به نظر برای تعداد ابعاد جدید ویژگی‌ها مناسب بود. اما با کاهش ابعاد ویژگی‌ها تا هفت هزار بعد، در مراحل بعدی حافظه‌ی colab در الگوریتمی چون svm کافی نبوده و crash کرد و برای جلوگیری از این مشکل به ناچار تعداد ابعاد را به حدود ۴۰۰ رساندیم. واضح است که این تصمیم‌گیری، سبب حذف برخی از ویژگی‌های مهم شده است.

لازم به ذکر است که با تعداد هفت هزار ویژگی الگوریتم knn اجرا شد و f1-score حدود ۱۸٪ حاصل شد. این مشکل می‌تواند حاصل از ذات الگوریتم knn نیز باشد؛ زیرا می‌دانیم که knn با ابعاد بالای ویژگی‌ها عملکرد خوبی ندارد. اما همان‌طور که ذکر شد امکان اجرای الگوریتم svm وجود نداشت وگرنه ممکن بود که این الگوریتم به خوبی با ابعاد بالاتر عمل کرده و نتایج بهتری حاصل شود.

عدم تصمیم‌گیری درباره‌ی برچسب‌ها

مشکل تساوی رأی در انتخاب برچسب نمونه‌ی جدید نیز یکی دیگر از مشکلات این مسئله است. در الگوریتم svm با استراتژی یک در مقابل بقیه، این مشکل با عنوان نواحی بدون تصمیم رایج است. اما این مشکل در سایر الگوریتم‌ها نیز مشاهده شد. این مشکل بدین صورت رخ می‌دهد:

قبل از ارسال برچسب‌ها به طبقه‌بند، ابتدا لازم است تا برچسب‌ها کد گذاری شوند تا برای الگوریتم قابل فهم باشند. بنابراین یک رشته‌ی ارقام به تعداد برچسب‌ها به دست می‌آید که هر رقم دو حالت دارد: صفر یا یک. اگر رقمی صفر باشد یعنی این نمونه به این کلاس تعلق ندارد و اگر یک باشد نشان تعلق نمونه به کلاس مورد نظر است. بدین ترتیب پس از فاز آموزش، نمونه‌های تست به همراه برچسب‌های صحیح به مدل آموزشی داده می‌شود و در نهایت برچسب‌های تخمینی از مدل به عنوان خروجی تحویل گرفته می‌شود. در حالت عدم تصمیم‌گیری این مورد پیش می‌آید که رشته‌ی برچسب باینری برابر با یک رشته‌ی بدون یک است و همه‌ی برچسب‌ها صفر هستند. یعنی الگوریتم نتوانسته برچسبی را برای نمونه‌ی دیده نشده پیدا کند. همین عامل سبب شده که برای برخی کلاس‌ها صحت و بازخوانی صفر داشته باشیم و در نتیجه f1-socre آن کلاس نیز صفر شده است. در نتیجه میانگین f1-score نیز به شدت پایین است.

به منظور حل این مشکل داده‌های آموزشی scale شد (به روش min max) تا میانگین و واریانس داده‌ها به ترتیب برابر صفر و یک شود و سپس طبقه‌بندی با الگوریتم‌ها نام برده انجام شد که این راه هم در حل مشکل تأثیری نداشته و همچنان مشکل عدم تصمیم‌گیری در کلاسِ نمونه وجود دارد.

منابع

Li, Y., Sheng, Y., Luan, L., & Chen, L. (2009). A Text Classification Method with an Effective Feature Extraction based on. *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*. IEEE.