گزارش پیاده سازی مقاله تشخیص کامنت های اسپم در یوتیوب

استاد درس: دکتر ایوب باقری

دانشجویان : سحر نفیسی ۹۳۲۱۱۷۰۰۲۶

علیرضا محمدی ۹۳۲۱۱۷۰۰۱۹

دانشگاه کاشان

زمستان ۹۶

مقدمه

پیاده سازی پیش رو بر اساس مقاله ی Tulio C. Alberto, Comment Spam Filtering on YouTube, IEEE, 2015 انجام شده است. در این مقاله تشخیص اسپم با روش هایی از جمله درخت تصمیم، ماشین بردار پشتیبان، شبکه های بیزین و ... انجام شده است. روشی که ما برای پیاده سازی برگزیدیم روش K نزدیکترین همسایه یا KNN است که با زبان پایتون و به کمک توابع کتابخانه ای nltk در پایتون انجام شده است.

پیشنیاز های اجرا و توضیحاتی در مورد طرز کار پروژه و کد

در اولین قدم، برای اجرای کد نیاز به پایتون نسخه 3 یا بالاتر داریم که از سایت رسمی پایتون قابل دریافت و نصب است. در ادامه لازم است لایبرری هایی که برای پیاده سازی پروژه از آن ها کمک گرفته شده است نصب شود. تنها کتابخانه ی آماده استفاده شده در این پروژه NLTK است که برای پیش پردازش متن کامنت ها (حذف علائم نگارشی، ریشه یابی، حذف کلمات زائد و ...) مورد استفاده قرار گرفته است. برای نصب این کتابخانه کافی است به اینترنت اتصال داشته باشیم و در خط فرمان دستور زیر را اجرا کنیم:

```
pip install nltk
```

در ادامه با استفاده از خط فرمان به پوشه ای که فایل `main.py` در آن قرار دارد می‌رویم و با استفاده از دستور زیر برنامه را اجرا می‌کنیم:

```
python main.py
```

اگر دفعه اول است که اسکریپت را اجرا می‌کنید، لازم است ۲ خط زیر را از کامنت خارج کنید:

```
nltk.download('punkt')
nltk.download('stopwords')
```

این دو تابع پس از اجرا، به روز ترین لیست از علائم نگارشی و کلمات زائد را از اینترنت دریافت می‌کنند، دفعات بعدی اجرا برای سرعت بیشتر، می‌توانید مجددا این دو خط را کامنت کنید.

هنگام اجرای برنامه لیست دیتاست هایی که در پوشه datasets قرار دارند نشان داده می‌شود و کاربر شماره یکی از آن ها را انتخاب می‌کند، سپس دیتاست مورد نظر از فایل خوانده می‌شود. در اینجا ما پس از خواندن دیتاست از فایل آن را به دو قسمت با نسبت های ۷۰ به ۳۰ تقسیم کریم، 70 درصد اول برای آموزش (training dataset) و ۳۰ درصد بعدی برای تست و ارزیابی (test case dataset) انتخاب می‌شوند. در ادامه بر روی دیتاست آموزشی پیش پردازش هایی صورت می‌گیرد (حذف کلمات زائد، ریشه یابی و ...) و لیست تمام کلمات به همراه فراوانی آن ها در کل دیتاست به دست می‌آید (Document Frequency) که با کمک آن مقدار tf-idf برای عبارات هر کامنت محاسبه می‌شود، برای تبیین بیشتر روند پروژه این اطلاعات را (لیست تمامی عبارات دیتاست و مقدار tf-idf برای هر عبارت در هر کامنت) را در فایلی موقتی به نام training_dataset_tfidf.csv دخیره کردیم (به این دلیل موقتی که پس از اجرای مجدد برنامه اطلاعات آن با توجه به دیتاست انتخابی بازنویسی می‌شود).

در ادامه مراحل پیش پردازش بر روی دیتاست تست انجام شده و سپس تمام آیتم های داخل دیتاست تست با الگوریتم KNN کلاسشان پیش بینی می‌شود. در اینجا نیز برای واضح تر شدن، نتایج را که شامل شناسه، متن، کلاس واقعی و کلاس پیش بینی شده توسط الگوریتم است را داخل فایلی موقتی به نام test_case_dataset_results.csv بر روی دیسک ذخیره کردیم. سپس درایه های ماتریس کانفیوژن را مقدار دهی کردیم و مقادیر Accuracy, Precision و ... را به دست آوردیم که در ادامه آن ها را تحلیل می‌کنیم.

خروجی پیاده سازی و نتایج به دست آمده

علی رغم تلاش های فراوان و بررسی های چندین و چند باره الگوریتم و مراحل پیش پردازش، نتیجه امیدور کننده ای از پیاده سازی حاصل نشد و نتایج فاصله زیادی نسبت به آن چه در مقاله مرجع وجود داشت، دارد. نکته جالب توجه دیگر که دلیلی برای آن پیدا نکردیم، انحراف از معیار بسیار بالای نتایج است، مثلا در معیار TN در دیتاست Psy مقدار 95 درصد به دست آوردیم ولی در دیتاست Skakira همین معیار، مقدار 28 درصد به دست آوردیم. مهم ترین دلیل این مسئله کم تجربگی ما در زمینه پردازش متن است و در ادامه دلایلی مانند استفاده نکردن از ابزار ها و کتابخانه های موجود و پیاده سازی الگوریتم از پایه، در اختیار نداشتن ابزار های پیش پردازش متن برای ریشه یابی و ... ممکن است در این موضوع تاثیر داشته باشند. در هر صورت نتایج به دست آمده بر روی ۳ دیتاست موجود به صورت زیر هستند:

Dataset	Accuracy	Precision	Recall	F-Measure	BH	TN
Psy	0.6226	0.9259	0.3968	0.55	0.6031	0.9534
LMFAO	0.8181	0.36	0.5294	0.4285	0.4705	0.8608
Shakira	0.5089	0.3888	1.0	0.56	0.0	0.2857