گزارش پروژه پایانی درس پردازش زبان طبیعی

استاد درس:

دکتر مهرنوش شمس‌فرد

دانشجویان:

تارا قشلاقی

فائزه آقازاده

نیمسال اول ۱۳۹۹-۱۴۰۰

hahackathon: Detecting and Rating Humor and Offense

hahackathons سیستم تشخیص لحن متن است. در این راستا تلاش می‌کند بتواند بطور خودکار طنز بودن/نبودن، و بحث‌برانگیز بودن را بررسی کند. منظور از بحث‌برانگیز بودن این است که برای متن نتوان با قاطعیت گفت طنز است. در ادامه و بعد از تشخیص طنز بودن متن، این برنامه یک امتیاز از ۰-۵ به میزان طنز بودن متن می‌دهد. از نوآوری‌های این پروژه بررسی توهین‌آمیز بودن متن است. به این منظور نیز برای هرمتن انتظار می‌رود سیستم یک امتیاز از ۰-۵ اختصاص دهد.

داده‌ها:

برای پروژه حاضر داده train و test موجود است. داده train شامل متن، لیبل برای طنز بودن، امتیاز طنز بودن، لیبل برای بحث‌برانگیز بودن، و درنهایت امتیاز توهین‌آمیز بودن است. داده تست اما فقط شامل متن است. در این پروژه به جای داده تست داده شده، از ۱۰٪ داده train بعنوان داده تست استفاده شده است.

مراحل اجرا:

برای اجرا پروژه در تمامی فازها از بستر google colab و استفاده شده و تمامی پردازش‌ها روی gpu انجام گرفته‌اند.

۱- Humor Detection

فاز اول این پروژه تشخیص طنز بودن متن است. به این منظور از مدل برت استفاده شده است. برت مدل مبتنی بر مفهوم transformerها است و برای پیکره‌های با حجم بالا بدون ناظر عملکرد خوبی از خود نشان می‌دهد. از دلایل استفاده از این مدل می‌توان به کتابخانه‌های موجود و سرعت بالای آن اشاره کرد. تعدادی از کتابخانه‌هایی که در این فاز استفاده شده‌اند عبارتند از:

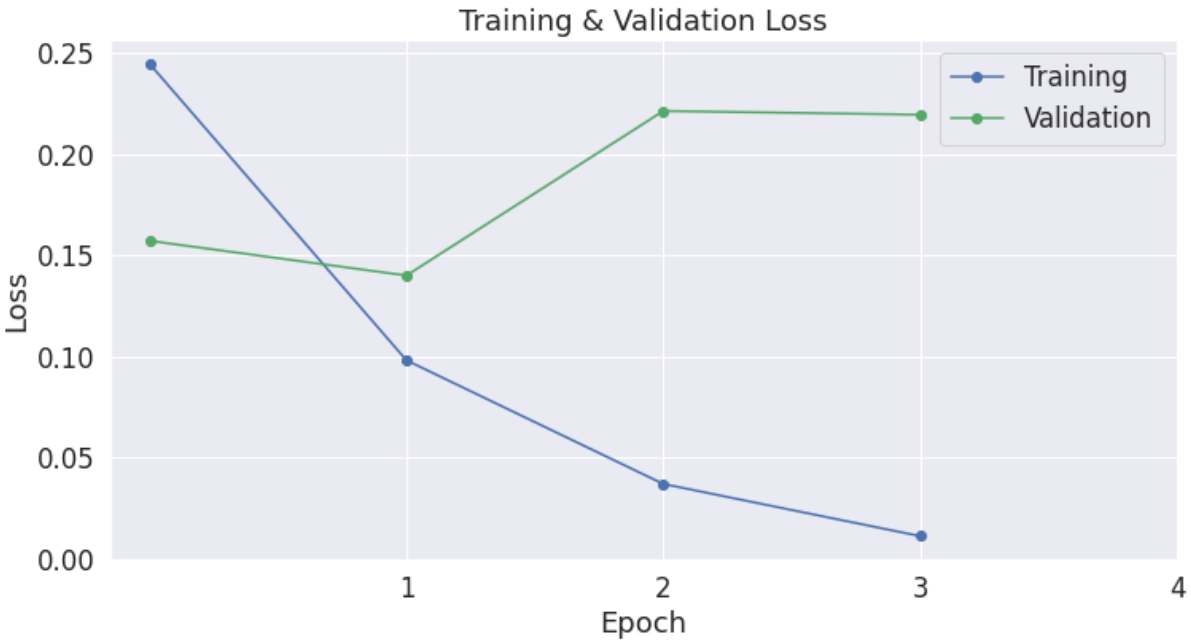
sklearn.model_selection: روشی برای تحلیل داده و سپس استفاده از این تحلیل برای پیش‌بینی داده‌های دیگر

torch.utils.data: برای استفاده از کتابخانه‌های دیگری همچون dataloader

tensorflow: برای محاسبات عددی و پردازش‌های عددی سنگین

در این مرحله فقط از متن و برچسب طنز بودن استفاده شده است. ستون‌های موجود در داده train، جداسازی شدند. بعبارتی متن‌ها در یک لیست و برچسب طنز بودن در یک لیست ذخیره شد. با توجه به اینکه اندازه متون یکسان نبود، maxlen تعیین کرده و متناسب با آن padding صورت گرفت. اندازه batch_size در این کد ۳۲ در نظر گرفته شده است. با توجه به اینکه این قسمت از کار یک binary classification است، از BertForSequenceClassification استفاده شده است. همچنین از بهینه‌ساز AdamW و تعداد تکرار ۴ استفاده شده است.

نتیجه train نشان می‌دهد که با کدهای موجود، بالاترین دقت برابر با ۹۵٪ است. همانطور که در پلات (شکل ۱) نیز قابل مشاهده است، در تست، با اینکه دقت نسبت خوبی در دور آخر بدست آمده است، ولی میزان loss نیز افزایش یافته است.



شکل ۱ . میزان loss برای فاز اول

نتایج هر epoch نیز در جدول یک نشان داده شده است.

Epoch	Accuracy	Training Loss	Validation Loss
1	0.94	0.25	0.16
2	0.95	0.10	0.15
3	0.94	0.03	0.23
4	0.95	0.01	0.24

جدول ۱ . نتایج ارزیابی فاز اول

مشاهده کد

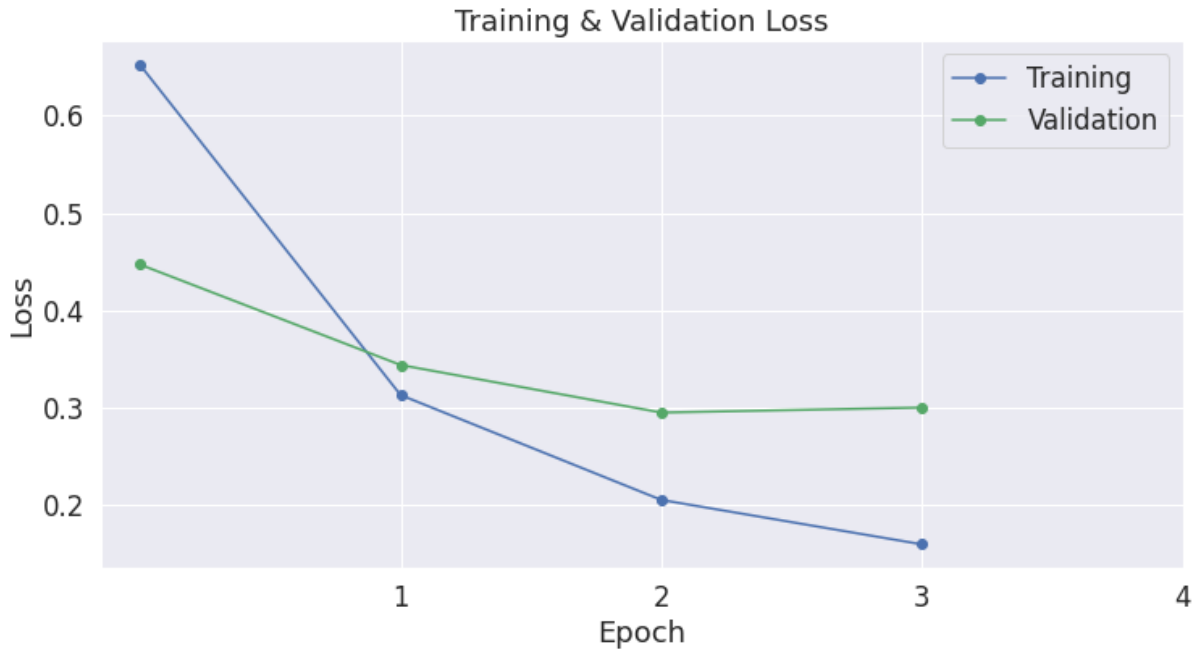
humor scoring -۲

هدف از این بخش، تعیین میزان طنز بودن متن (درصورت طنز بودن آن) است. این میزان عددی بین ۰-۵ است.

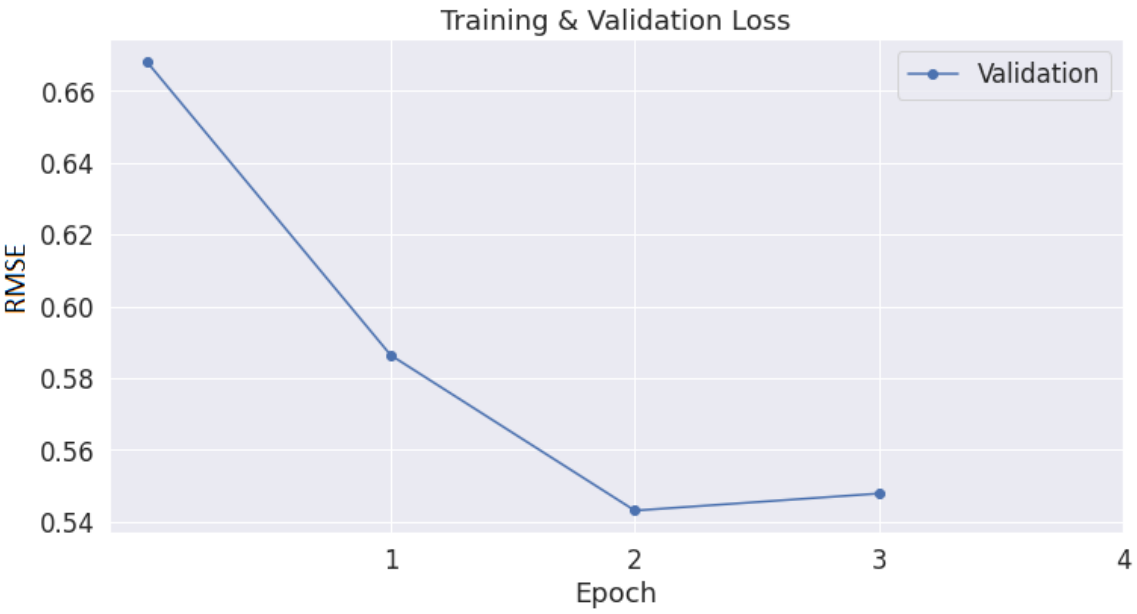
برای پیاده‌سازی این بخش، بسیاری از کارهای پیش‌پردازش فاز قبلی تکرار شده‌اند که از بیان مجدد آنها پرهیز می‌شود.

در این بخش مجددا از مدل برت استفاده شده است. خروجی مدل برت یک عدد float در بازه موردنظر ما است. همانطور که در پلات اول (شکل ۲) نیز قابل مشاهده است، میزان loss با هر تکرار کاهش می‌یابد. در این بخش متریک RMSE هم محاسبه

شده است. پلات دوم (شکل ۳) متریک RMSE را نشان می‌دهد که با هر تکرار کمتر می‌شود. بهترین مقدار برای RMSE برابر با ۰,۵۴ بدست آمده است.



شکل ۲ . میزان loss



شکل ۳ . میزان RMSE در تکرارهای مختلف

همچنین، در جدول ۲ می‌توانید جزئیات ارزیابی را در ۴ تکرار ببینید.

Epoch	RMSE	Training Loss	Validation Loss
1	0.67	0.65	0.45
2	0.59	0.31	0.34
3	0.54	0.21	0.30
4	0.55	0.16	0.30

جدول ۲ . نتایج ارزیابی فاز دوم

مشاهده کد

۳- scoring offensive

هدف از این بخش، تشخیص میزان توهین‌آمیز بودن متن است. در این قسمت اهمیتی به طنز بودن یا نبودن متن داده نمی‌شود. فرض بر این است که متن چه طنز باشد چه نباشد می‌تواند توهین‌آمیز باشد. تعدادی از عملیات این قسمت مشابه فاز قبلی است، بنابراین از تکرار آنها صرفنظر می‌شود.

برای تعیین میزان توهین‌آمیز بودن، علاوه بر اقدامات فاز قبلی از iob tagging استفاده شده است. برنامه‌نویسان این پروژه بر این باور بودند که چنانچه اسامی خاص مانند نام سازمان، شخص خاص و ... در متن باشد می‌توان بعنوان فیچری برای توهین‌آمیز بودن درنظر گرفت.

کار دیگری که در این بخش انجام گرفته است، استفاده از معیار tfidf برای تعیین میزان توهین‌آمیز بودن است. این مفهوم به ما اجازه می‌دهد براساس کلمات موجود در متن تصمیم بگیریم که متن توهین آمیز است یا نه، و به چه میزان توهین‌آمیز است. در این راستا، کلمات به بردار تبدیل شدند و پردازش‌هایی مانند tokenizing و حذف stopwordها انجام شده است.

در ادامه از svm و رگرسیون استفاده شده است. Svm از الگوریتم‌های تشخیص الگو است. در این بخش نیز به همین منظور استفاده شده است. درواقع با این دو، می‌خواهیم به یک الگو برای توهین‌آمیز بودن متن برسیم، و سپس براساس آن بگوییم متون ورودی تا چه حد توهین‌آمیز هستند. مشابه فاز قبل، متریک ارزیابی شده در این بخش RMSE است که مقدار آن برابر با ۰٫۵۷ شده است.

مشاهده کد

۴- humor controversy

هدف از این بخش تعیین این مسئله است که آیا متن ورودی بحث‌برانگیز است یا نه. مشابه فاز اول یک تسک دوکلاسه است.

در این بخش نیز پیش‌پردازش‌های مشابه تکرار شده‌اند. همچنین با دستور CountVectorizer یک vocabulary ساخته شده است که برای متون ورودی از آن استفاده می‌شود. روند کار بسیار شبیه به فاز قبلی است. از tfidf و svm برای تعیین controversy استفاده شده است. نکته‌ای که در این بخش اضافه شده، استفاده از لیبل‌های humor score و offensive

score برای ورودی svm است. نتایج ارزیابی این برای دو کلاس ۰ و ۱ (controversy بودن یا نبودن) در جدول ۳ نشان داده شده است. دقت در این مرحله ۷۰٪ بدست آمده است. در جدول ۴ نیز نتایج کلی نمایش داده شده است.

class	Precision	Recall	F1-Score
0	0.74	0.88	0.80
1	0.50	0.28	0.36

جدول ۳ . نتایج ارزیابی فاز چهار

Accuracy	Precision	Recall
0.70	0.50	0.28

جدول ۴ . نتایج کلی ارزیابی

مشاهده کد

بخش‌های اضافه شده

شرح داده:

Train

داده train شامل ۵ ستون به ترتیب شامل ۱- متن ۲- برچسب تشخیص طنز بودن (صفر و یک) ۳- میزان طنز بودن (عدد اعشاری بین صفر و پنج) ۴- برچسب دوپهلو بودن (صفر و یک) ۵- میزان توهین‌آمیز بودن (عدد اعشاری بین صفر و پنج)

Test

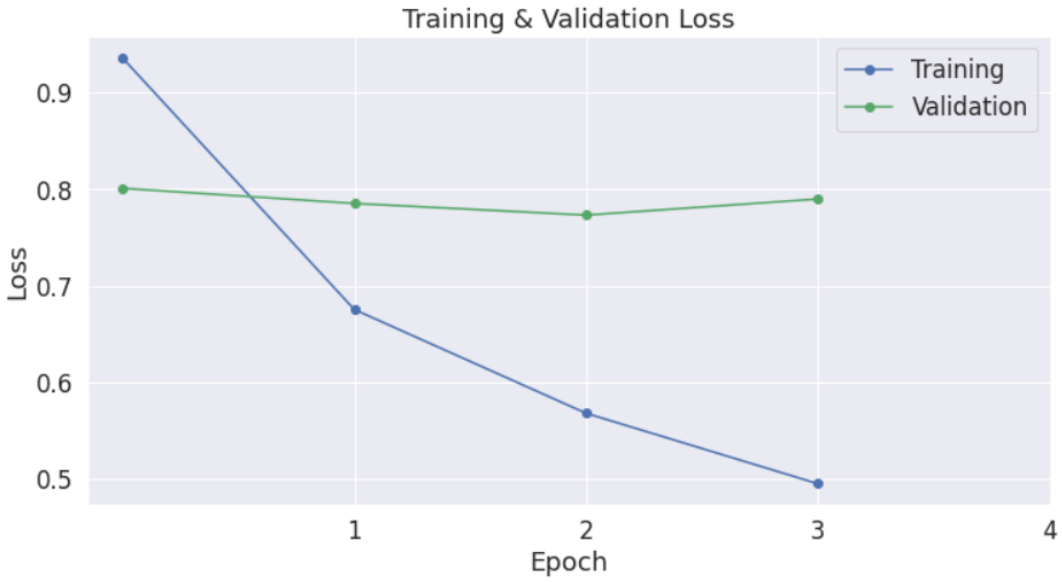
داده تست فقط شامل متن است، و برچسبی ندارد.

کلاس‌بندی humor scoring

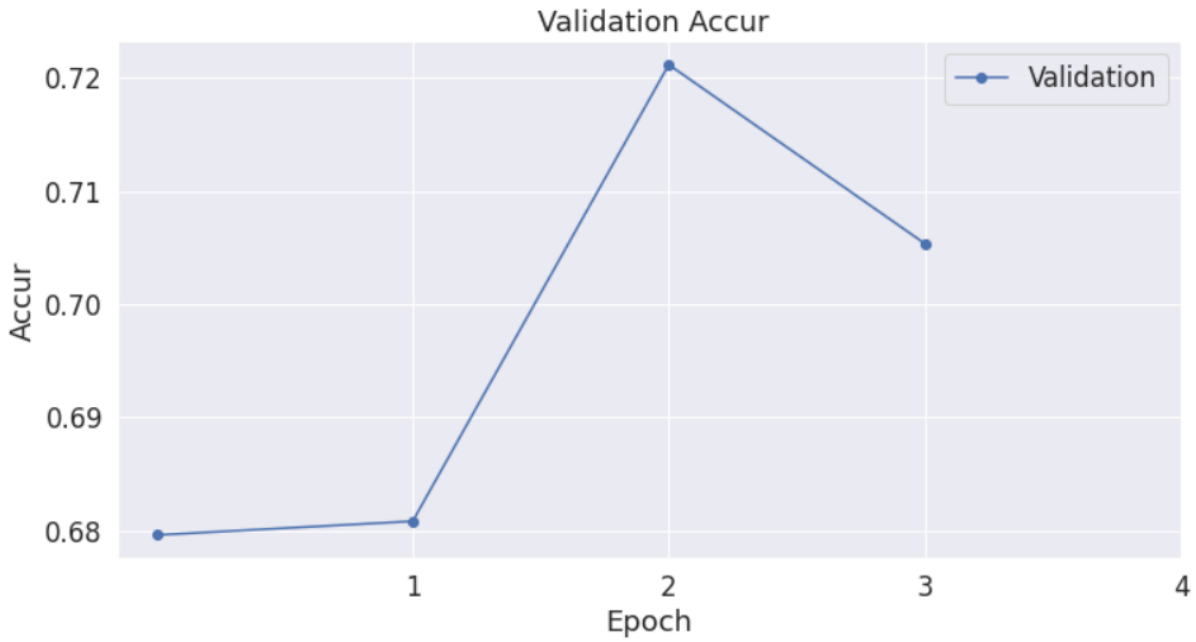
در بخش قبلی، امتیازدهی به شکل عدد اعشاری بود. در این بخش اعداد اعشاری گرد و به یکی از ۶ کلاس از ۰-۵ اختصاص داده شده‌اند. همچنین، punctuation marks هم از متون حذف شدند. نتیجه این تغییر در جدول ۵ و اشکال ۴ و ۵ قابل مشاهده است.

Epoch	Accuracy	Training Loss	Validation Loss
1	0.68	0.94	0.80
2	0.68	0.68	0.79
3	0.77	0.57	0.72
4	0.71	0.49	0.79

جدول ۵. نتایج ارزیابی (با کلاس‌بندی)



شکل ۴. میزان loss در چهار تکرار



شکل ۵. میزان accuracy در چهار تکرار

مشاهده کد

کلاس‌بندی offensive scoring

در این بخش نیز مشابه قبلی، میزان توهین‌آمیز بودن را به ۶ کلاس از ۰ تا ۵ دسته‌بندی کرده‌ایم. نتیجه این دسته‌بندی در جدول ۶ نشان داده شده است. همچنین جدول ۷ نیز یک ارزیابی کلی را نشان می‌دهد.

Class	Precision	Recall	F1-Score
0	0.77	0.99	0.86
1	0.35	0.09	0.14
2	0.28	0.10	0.15
3	0.29	0.05	0.09
4	0.59	0.24	0.34
5	0.0	0.0	0.0

جدول ۶. نتایج کلاس‌بندی offensive scoring

Accuracy	Precision	Recall
0.73	0.65	0.73

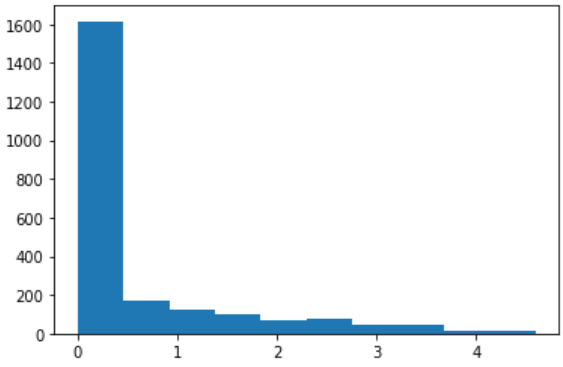
جدول ۷. نتیجه ارزیابی کلی

مشاهده کد

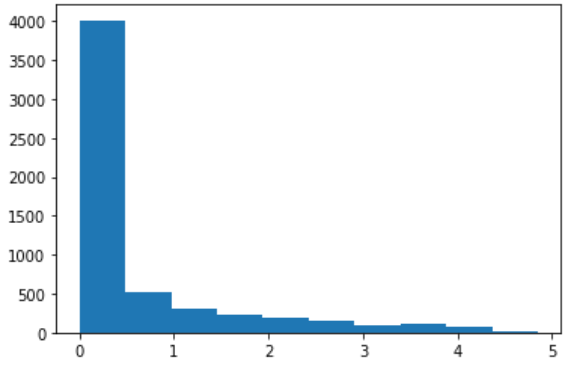
حذف NER و بخش iob tagging

برای مقایسه اثربخشی NER، و با توجه به اینکه قبلا این بخش لحاظ شده بود، در این قسمت NER را از کد حذف کردیم تا ببینیم در غیاب آن دقت برنامه چه تغییری می‌کند. نتیجه اجرای این بخش نشان می‌دهد وجود/عدم وجود NER تاثیری در عملکرد برنامه ندارد. مقدار RMSE در این بخش برابر با ۰,۵۷ است که همان مقداری است که در اجرا با NER بدست آمده بود.

همچنین برای نمایش شهودی عدم تاثیر NER، فراوانی میزان offensive بودن رسم شده است. شکل ۶ فراوانی میزان توهین‌آمیز بودن در حضور NER و شکل ۷ بدون NER هست. همانطور که در شکل‌های زیر مشاهده می‌شود، فراوانی score تغییری نمی‌کند.

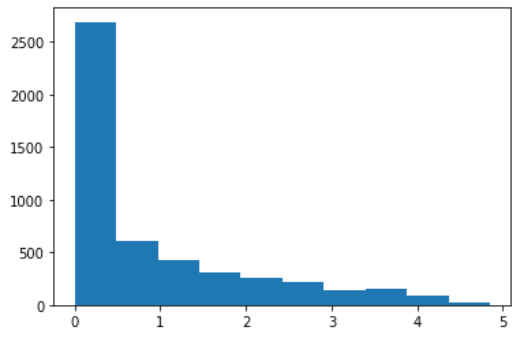


شکل ۶ . فراوانی offensive scoring با NER

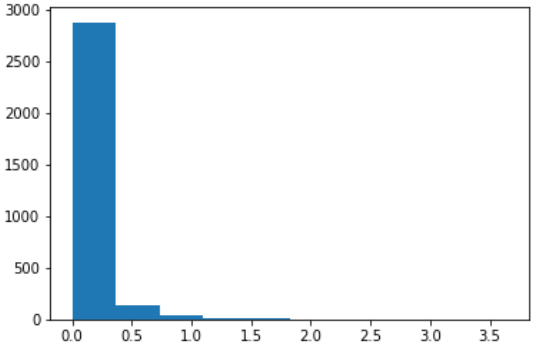


شکل ۷ . فراوانی offensive scoring بدون NER

همچنین، علاوه بر NER، اثربخشی برچسب is humor (طنز بودن یا نبودن) در میزان Offensive نیز بررسی شده است. اشکال ۸ و ۹ میزان توهین‌آمیز بودن با درنظر گرفتن is humor و بدون درنظر گرفتن آن را نشان می‌دهند. باتوجه به شکل‌های زیر می‌توان گفت is_humor فیچر خوبی برای تشخیص میزان offensive بودن است.



شکل ۹ . فراوانی offensive scoring با is_humor



شکل ۸ . فراوانی offensive scoring بدون is_humor

مشاهده کد