به نام خدا

naïve bayes classifier

تهیه کننده : عاطفه محمدی

توضیح کلی صورت پروژه

در این پروژه بنا داریم تا عمل sentiment analysis را بر روی کامنت هایی روی کالاهایی از دیجی کالا انجام دهیم؛ بدین ترتیب که پیش بینی کنیم یک کامنت به خصوص، کالای مورد نظر را پیشنهاد کرده است یا خیر. به این منظور دو dataset در فرمت csv در اختیار ما قرار گرفته است که اولی مجموعه داده train و دومی مجموعه داده test ما هستند. هر دو مجموعه داده train و test دارای سه ستون هستند؛ عنوان کامنت، متن کامنت و ستون هدف که نمایانگر مثبت یا منفی بودن کامنت است. مسئله از نوع supervised learning و classification است چرا که در مجموعه داده train، لیبل های کامنت ها شامل recommended و not recommended می باشد و مشخص است.

فاز اول: پیش پردازش داده

مشخص است که پیش از آن که داده را به عنوان ورودی به مدل classifier بدهیم، باید پیش پردازش هایی را روی آن انجام دهیم. بررسی اولیه روی مجموعه داده train نشان می دهد که هیچ مقدار Null ای در آن وجود ندارد و مجموعه داده متوازن است. به جهت پردازش اولیه مجموعه داده train به این ترتیب عمل می کنیم که سطر به سطر دیتافریم را می خوانیم و نوشته های موجود در دو ستون اول آن را استخراج می کنیم زیرا رویکرد ما در این پیاده سازی این است که عنوان و متن هر کامنت را مجموعا یک نوشته در نظر می گیریم و عملا به هر دو وزن یکسانی تخصیص می دهیم. سپس ابتدا ایموجی ها را از این نوشته ها حذف می کنیم. بعد از آن به کمک کتابخانه hazm، متن ها را normalize و سپس بر حسب کلمات آن ها را توکن بندی می کنیم. در ادامه stop words را از میان این کلمات حذف می کنیم (stop words به کار رفته در این پیاده سازی، شامل stop words کتابخانه hazm و نیز تعدادی stop words اضافه شده به صورت دستی می باشد). منظور از stop words کلماتی است که ارزش پردازش کردن را ندارند؛ مثلا اعداد، حروف اضافه و علائم نگارشی روی پیشنهاد شدن یا نشدن یک کالا بی تاثیرند.

در نهایت، ریشه کلمات را استخراج کرده و آن ها را به bag of words اضافه می کنیم. bag of words مجموعه ای است که بدون توجه به جایگاه کلمات در جمله ها، تمام کلمه های مفید در train dataset را در خود ذخیره کرده است. سپس از روی آن یک دیتافریم جدید می سازیم که ستون های آن همان کلمات bag of words هستند و مقادیر موجود در هر سطر نمایان گر تعداد دفعات تکرار آن کلمه در هر کامنت است.

```
stmr = Stemmer()
nrmlzr = Normalizer()
bag_of_words_train = set()
corpus = []
for row in range(train_data.shape[0]):
    data = dict()
    for col in ['title', 'comment']:
        sent = train_data.loc[row, col]
        sent = emoji_removing(sent)
        sent = nrmlzr.normalize(sent)
        words = word_tokenize(sent)
        tmp = []
        for word in words:
            if word not in stopwords_list() and word not in stop_words:
                tmp.append(word)
        for t in tmp:
            s = stmr.stem(t)
            bag_of_words_train.add(s)
            if s in data.keys():
                data[s] += 1
            else:
                data[s] = 1
    data['recommend'] = train_data.loc[row, 'recommend']
    corpus.append(data)   # making a list of dictionaries
```

1) در فاز ارزیابی پروژه، در قسمتی که فقط از پیش پردازش داده استفاده می کنیم، به جای stemmer کتابخانه هضم، از lemmatizer نیز استفاده شده است و تاثیر آن ها در نتیجه با هم مقایسه گردیده است. تفاوت این دو تابع در شکل زیر مشخص شده است.

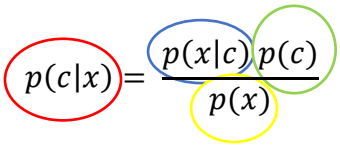
```
>>> from hazm import Stemmer, Lemmatizer
>>> stemmer = Stemmer()
>>> stemmer.stem('کتاب‌ها')
'کتاب'
>>> lemmatizer = Lemmatizer()
>>> lemmatizer.lemmatize('می‌روم')
'رفت#رو'
```

فاز دوم: فرایند مسئله

2)

کلاس recommended را معادل ۱ و کلاس not recommended را معادل ۰ در نظر می گیریم.

با توجه به قانون بیز داریم:



$$p(c|x) = \frac{p(x|c)\,p(c)}{p(x)}$$

عبارت اول (دایره قرمز) یا posterior probability به معنی احتمال کلاس c (recommended/not recommended) برای کامنت داده شده ی x است. این همان احتمالی است که به دنبال پیدا کردنش هستیم.

عبارت دوم (دایره آبی) یا likelihood مفهوم برعکس عبارت اول را دارد. یعنی برای کلاس داده شده ی c، چقدر احتمال دارد که کامنت x به این کلاس متعلق باشد. نحوه محاسبه این مورد در ادامه گفته می شود.

عبارت سوم (دایره سبز) احتمال پیشامد هر کدام از کلاس های recommended/not recommended در مجموعه داده train است که چون مجموعه داده train ما متوازن بود، احتمال هر دو آن ها برابر 0.5 می باشد.

مخرج کسر (دایره زرد) احتمال هر کدام از کامنت هاست؛ یعنی احتمال رخداد هر نوشته صرف نظر از این که به کدام کلاس متعلق است، که چون برای هر دو کلاس یک عدد ثابت است (برای این که حاصل کسر بین ۰ و ۱ باشد)، و ما فقط قصد مقایسه دو کلاس را داریم و عدد دقیق آن برایمان اهمیت ندارد، آن را نادیده می گیریم.

در مرحله پیش پردازش، ما متن کامنت را با یک بردار عددی که هر مولفه آن برابر یک لغت از bag of words بود، جایگزین کردیم. بنابراین هر کامنت، به مجموعه ای از کلمات تبدیل شده است:

$$x = (x_1, x_2, \dots, x_n)$$

با فرض مستقل بودن $p(x_i|c_j) \quad i \in [1, \dots, n], j \in [0,1]$ ها از همدیگر، عبارت بالا را به صورت زیر خلاصه می کنیم:

$$p(c|x) = p(c) \prod_{x \in [1,..,n]} p(x_i|c)$$

یعنی عبارت likelihood را به صورت حاصل ضرب احتمال شرطی تک تک کلمات تشکیل دهنده هر کامنت، به شرط وقوع هر کلاس تغییر دادیم. پس به ازای هر کامنت در مجموعه داده test، دو احتمال به دست می آوریم که یکی احتمال اختصاص این کامنت به کلاس ۰ و دیگری احتمال اختصاص آن به کلاس یک است.

برای محاسبه $p(x_i|c)$ها، از bag of words مجموعه داده train استفاده می کنیم و برای هر کلاس بخصوص (۰ یا ۱)، تعداد دفعات تکرار لغت x_i در آن کلاس را تقسیم بر تعداد کل کلمات در آن کلاس می کنیم.

additive smoothing :

(3

فرض کنیم : «همه محصولات دیجی خوب است» = c . c کامنتی از مجموعه داده test است که در حال محاسبه احتمال اختصاص آن به کلاس ۰ می باشیم. مثلا اگر کلمه «دیجی» در bag of words کلاس صفر در مجموعه داده train موجود نباشد، احتمال آن صفر به دست می آید. و چون کل عبارت از حاصل ضرب تشکیل شده است، احتمال اختصاص c به کلاس ۰، صفر در می آید.

یعنی چون لزوما bag of words در مجموعه داده train با bag of words در مجموعه داده test یکی نیست، عدم حضور کلمات یکی در دیگری موجب صفر شدن برخی احتمالات شده و پیش بینی مدل ما را مختل خواهد کرد.

(4

یک راه برای حل این مشکل استفاده از additive smoothing می باشد. این روش به مقدار همه x_iها یک عدد ثابت لاندا (معمولا ۱) اضافه می کند؛ بنابراین به صورت $p(x_i|c)$ ها یک اضافه می شود و به مخرج آن ها n+1 اضافه می گردد.

فاز سوم: ارزیابی

(5

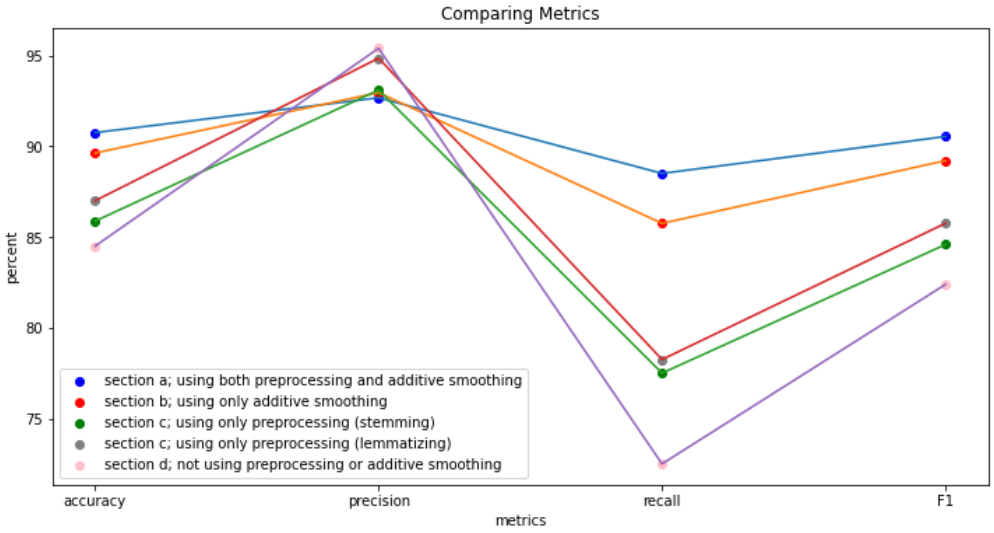
precision نشان می دهد چه تعداد از آیتم های انتخاب شده توسط الگوریتم، مربوط (درست) هستند. recall یعنی از مقادیر درستی که باید همه اش توسط الگوریتم انتخاب می شده، چه کسری انتخاب شده است.

ضعف هر یک از آن ها به تنهایی آن جا مشخص می شود که تعداد false positive ها یا false negative ها صفر باشد یا خیلی کم باشد. مثلا اگر false positive نداشته باشیم مقدار precision یک می شود و اگر false negative نداشته باشیم، مقدار recall یک خواهد شد. در حالی که مدل لزوما درست کار نمی کند.

(6

معیار f1 در واقع میانگین هارمونیک دو معیار precision و recall است. چرا که precision و recall دو کسر با صورت های مساوی و مخرج های متفاوت هستند؛ بنابراین تنها نوع میانگین که منطقا میان این دو می توان گرفت، میانگین هارمونیک است. مثل این که بخواهیم سرعت متوسط متحرکی را در دو بار رفتن از نقطه «الف» به «ب» با دو سرعت متفاوت، محاسبه کنیم.

(7



8) با توجه به شکل بالا مقدار precision در هر ۵ حالت نزدیک هم است. اما در مورد سه معیار دیگر ترتیب زیر از بیشترین به کمترین دیده می شود:

۱- a : استفاده از پیش پردازش و additive smoothing

۲- b : فقط استفاده از additive smoothing

۳- c : فقط استفاده از پیش پردازش داده به روش lemmatizing

۴- c : فقط استفاده از پیش پردازش داده به روش stemming

۵- d : استفاده نکردن از هیچ کدام

9) ۵ مورد از کامنت هایی که در حالت a اشتباه برچسب خورده اند، در نوت بوک در قسمت a آمده است.

منابع:

/https://machinelearningmastery.com/gentle-introduction-bag-words-model

https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

https://www.datacamp.com/community/tutorials/simplifying-sentiment-analysis-python

https://towardsdatascience.com/unfolding-na%C3%AFve-bayes-from-scratch-2e86dcae4b01