به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق وکامپیوتر

# درس پردازش زبان های طبیعی

## پروژه شماره : 2

**نام و نام خانوادگی : محمدرضا بختیاری**

**شماره دانشجویی : 810197468**

فروردین 1401

## فهرست سوالات

## پیش پردازش ها

پیش از تولید یک مدل زبانی , ابتدا نیاز داریم متناسب با هدف طبقه بند پیش پردازش های لازم را انجام دهیم.

ابتدا نمونه هایی از پیش پردازش های مرسوم در پردازش زبان های طبیعی را مشاهده می کنیم و سپس متناسب با هدف نهایی هر بخش , پیش پردازش های مناسب هر بخش را انتخاب و پیاده سازی می کنیم.

نمونه هایی از پیش پردازش های مرسوم :

Lowercasing - Removing Extra Whitespaces – Tokenization - Spelling Correction - Stopwords Removal - Removing Punctuations - Removal of Frequent Words – Stemming – Lemmatization - Removal of URLs - Removal of HTML Tags

توجه شود در این قسمت ملزم به استفاده از تمامی این پیش پردازش ها نیستیم و تنها از آنهایی استفاده می کنیم که توجیه کننده هدف نهایی ما باشد. به عنوان مثال در تشخیص هرزنامه بودن[1] یا نبودن یک پیام , پیش پردازش Lowercasing ممکن است منجر به از دست رفتن اطلاعات شود ( یکی از قوانین دستی[2] برای تشخیص اسپم بودن یک پیام می تواند این باشد که در پیام های هرزنامه , از حروف بزرگ زیاد استفاده می شود ) یا همان طور که در صورت پروژه اشاره شد استفاده از پیش پردازش Removing Punctuations امکان تشخیص اسپم بودن با وجود "!" از بین می برد.

همچنین پیش پردازش هایی اعم از Removal of HTML Tags , Removal of URLs , ... نیز در تشخیص اسپم بودن پیام کاربردی نخواهند داشت و حتی ممکن است منجر به از دست رفتن اطلاعات مفید شود. ( وجود لینک و تگ های HTML می تواند نشانه ی خوبی برای اسپم بودن یک پیام باشد. )

در اینجا در تسک مربوط به تشخیص اسپم بودن پیام , Tokenization , Spelling Correction و Stemming می توانند گزینه های مناسبی برای پیش پردازش باشند که ما از Tokenization استفاده می کنیم.

برای تسک مربوط به تشخیص راست یا دروغ بودن پیام نیز Tokenization گزینه مناسبی برای پیش پردازش می باشد.

پ . ن : در ادامه خواهیم دید که به جای استفاده از Tokenization می توانیم از CountVectorizer() استفاده کنیم که همان غایت نهایی Tokenization را به ارمغان می آورد و منجر به تشکیل ماتریس document-term می شود.

[1] Spam detection
[2] Hand-coded rules

# استخراج ویژگی ها

## دیتاست UCIML

برای تشخیص اسپم بودن پیام می توانیم ویژگی های متعددی در نظر بگیریم. به عنوان مثال Hand-coded rule های متعددی برای این تسک وجود دارد و می توانیم از آن ها استفاده کنیم که در اینجا به چند مورد از آن ها اشاره می کنیم :

وجود علامت هایی نظیر "$" یا نقطه گذاری هایی مثل "!" , تکرر استفاده از حروف بزرگ , استفاده از URL ها و لینک ها و هشتگ های HTML , استفاده از black-list-address ها و یا وجود عباراتی نظیر "have been selected" همگی می توانند ویژگی های مناسبی برای اسپم بودن پیام باشند.

به علت time-consuming بودن و متغیر بودن این قوانین به واسته ی اضافه شدن قوانین جدید در هر زمان ما در ادامه فقط تکرر هر لغت را در نظر می گیریم و احتمال متعلق بودن هر لغت به کلاس مورد نظر (spam – ham) را محاسبه کرده و از الگوریتم naive bayes برای طبقه بند خود استفاده می کنیم و مشاهده می کنیم که به دقت مناسبی (حدود 97.6) دست پیدا کرده ایم.

## دیتاست Sentimental LIAR

برای تشخیص راست یا دروغ بودن نیز همان طور که در صورت سوال اشاره شده می توانیم از ستون sentiment و 5 ستون anger , fear , joy , disgust , sad که نرخ احساسات را نشان می دهند به عنوان ویژگی در آموزش طبقه بند استفاده کنیم.

ابتدا نیاز داریم مقادیر ستون sentiment را به صورت عددی دربیاوریم. با توجه به این که ما کلاس False را معادل 0 و کلاس True را معادل 1 در نظر گرفتیم , می توانیم NEGATIVE را معادل با 0 و POSITIVE را معادل با 1 در نظر بگیریم. همچنین سطر هایی که sentiment_score یا sentiment_magnitude آن ها معادل با 0 است , sentiment آن ها معادل با Null است که می توانیم به راحتی این سطر ها را حذف کنیم.

همچنین 5 ستون دیگر نیز مقادیری بین 0 و 1 دارند که نیاز داریم این مقادیر را باینری کنیم. به این صورت که با تعریف یک آستانه[1] مقادیر بالاتر از این آستانه را معادل با 1 و سایر مقادیر را معادل با 0 در نظر بگیریم. این آستانه را می توانیم به صورت یک فراپارامتر[2] در نظر بگیریم و مقدار آن را محاسبه کنیم. (در اینجا مقدار 0.25 برای آستانه دقت خوبی به ما می دهد.)

[1] Threshold
[2] Hyperparameter

## آموزش طبقه بند ها

در ادامه با استفاده از Multinomial naive bayes classifier طبقه بند خود را آموزش می دهیم. به این صورت که با توجه به احتمالات به دست آمده در قسمت قبل و شرط مهم استقلال این طبقه بند داده متعلق به کلاسی است که احتمال زیر را بیشینه کند :

$$C_{NB} = argmax\, P(c_i) * \prod P(w|c)$$

که در آن :

$$P(c_j) = \frac{count(C = c_j)}{N_{doc}}$$

همچنین برای محاسبه ی احتمال $P(w_i|c)$ و جلوگیری از صفر شدن احتمال کل می توانیم از Laplace smoothing استفاده کنیم :

$$P(w_i|c) = \frac{count(w_i, c) + \alpha}{count(w, c) + \alpha * V}$$

که در اینجا V معادل است با تعداد کل کلمات موجود در document و $\alpha$ را نیز می توانیم به عنوان یک هایپر پارامتر در نظر بگیریم و مقدار آن را محاسبه کنیم. (به عنوان مثال $\alpha$ برابر با یک معادل است با add-1)

در اخر با استفاده از MultinomialNB() آموزش مدل خود را تکمیل می کنیم.

# ارزیابی

## دیتاست UCIML

در نهایت پس از آموزش مدل , عملکرد مدل مورد نظر را بر روی دادگان تست , ارزیابی می کنیم و معیار های خواسته شده را گزارش می کنیم.

```
Confusion matrix is =
 [[3840   18]
 [  89  511]]
 ----------
Accuracy is =
 97.59982054733064  %
 ----------
Recall is =
 0.9773479256808348
 ----------
Precision is =
 0.995334370139969
 ----------
F1 measure is =
 0.98625914986516
```

*شکل 1 : معیار های ارزیابی برای دادگان UCIML*

$Accuracy = 97.599\ \%$ $Recall = 0.977$

$Precision = 0.995$ $F1\ measure = 0.986$

مشاهده می کنیم طبقه بند از دقت مناسبی برخوردار بوده و باتوجه به ماتریس آشفتگی[1] , طبقه بند بیشتر در مواردی که پیام spam بوده و به اشتباه به عنوان ham شناسایی شده دچار خطا شده است.

حال برای این که بررسی کنیم ویژگی های ذکر شده در قسمت استخراج ویژگی برای تشخیص اسپم بودن پیام , چقدر روی بهبود عملکرد نهایی طبقه بند تاثیر می گذارد , چند نمونه از داده هایی که در اصل spam بوده اند و به اشتباه ham تشخیص داده شده اند (false negatives) را بررسی می کنیم.

```
You have 1 new voicemail. Please call 08719181...
Reminder: You have not downloaded the content ...
LookAtMe!: Thanks for your purchase of a video...
 You have 1 new message. Please call 08718738034.
Send a logo 2 ur lover - 2 names joined by a h...
                      ...
Please CALL 08712402779 immediately as there i...
U were outbid by simonwatson5120 on the Shinco...
Free msg. Sorry, a service you ordered from 81...
Sorry I missed your call let's talk when you h...
Someone U know has asked our dating service 2 ...
```

*شکل 2 : چند نمونه از پیام هایی که به اشتباه ham تشخیص داده شده اند.*

[1] Confusion matrix

همانگونه که مشاهده می کنیم اگر استفاده از عباراتی مثل "call" و یا وجود شماره تلفن را به عنوان یک ویژگی برای اسپم بودن پیام در نظر می گرفتیم تعداد عباراتی که به اشتباه ham تشخیص داده شده اند کاهش پیدا می کرد و دقت باز هم افزایش پیدا می کرد.

اما همانگونه که در قسمت استخراج ویژگی اشاره شد , استفاده از Hand-coded rules ها علاوه بر این که زمان بر و پرهزینه است , ممکن است این قوانین در طول زمان دچار تغییر شوند و نیاز باشد که دوباره قوانین جدیدی وضع کنیم پس امروزه استفاده از این روش ها دیگر کارآمد[1] نخواهد بود.

## دیتاست Sentimental LIAR

در این قسمت نیز همانند بخش قبلی عملکرد مدل را بر روی دادگان تست بررسی می کنیم و معیار های خواسته شده را گزارش می کنیم.

```
Confusion matrix is =
 [[263 290]
 [202 512]]
 ----------
Accuracy is =
 61.16811365430151  %
 ----------
Recall is =
 0.5655913978494623
 ----------
Precision is =
 0.4755877034358047
 ----------
F1 measure is =
 0.5166994106090373
```

*شکل 3 : معیار های ارزیابی برای دادگان Sentimental LIAR*

$Accuracy = 61.168\,\%$ $Recall = 0.565$

$Precision = 0.475$ $F1\ measure = 0.516$

همان گونه که مشخص است مدل عملکرد مناسبی نداشته و دقت آن عملا تفاوت چندانی با حدس زدن کلاس خروجی به صورت تصادفی ندارد. (حتی در صورت استفاده از برخی ویژگی های ذکر شده , دقت مدل کمتر هم خواهد شد (نزدیک به 57 درصد)).

[1] Efficient

## پیوست:

تمامی فایل های ipynb در پوشه ی Codes موجود می باشد. همچنین تمامی کد ها در محیط colab اجرا و تست شده اند.

**توجه** : نیاز است در هنگام اجرای برنامه ها , دیتاست های لازم برای هر بخش ( SMSSpamCollection برای دیتاست اول و train_final و test_final برای دیتاست دوم ) دقیقا در دایرکتوری در کنار کد قرار گیرند.