شرح کلی پروژه

در این پروژه با پردازش متن و چگونگی استفاده از قاعده بیزین برای به دست آوردن دسته بندی یک خبر آشنا شدیم. در ادامه روند کلی کار را توضیح می دهیم.
برای پیش بینی دسته بندی یک خبر، اولین گام فراهم آوردن یک لغت نامه از واژه ها و کلمات مربوط به هر دسته بندی مورد مطالعه می باشد.
روش به کار گرفته شده در این پروژه، bag of words نام دارد که طی آن با استفاده از تعداد مطلوبی خبر که دسته بندی آن ها را قبلا می دانیم، یک لغت نامه از کلمات مربوط به دسته بندی های مختلف مهیا کرده و دسته بندی داده های (خبر های) جدید را با استفاده از این لغت نامه و مدل به دست آمده تعیین می کنیم.

یک فایل data.csv برای آموزش و تهیه و ارزیابی مدل و یک فایل test.csv برای تعیین دسته بندی تعدادی خبر مجهول داریم. ابتدا مدل را با استفاده از data.csv آماده کرده و بعد از ارزیابی آن، خبر های فایل test.csv را با این مدل تعیین می کنیم.

در گام نخست، خبرهای فایل data.csv که با ستون short description شناسایی می شوند و برای فاز آموزش و ارزیابی مدل به کار می روند را پردازش می کنیم.

برای پیش پردازش داده لازم است که آن را از جهات مختلف از جمله حذف کلمات تکراری و پر کاربرد انگلیسی (مانند and ،the و...)، تبدیل همه حروف به lower case، حذف علائم نگارشی (مانند !،>،< و دیگر علامت ها که با کتابخانه string به دست می آیند) و stem یا lemmatization فیلتر کنیم که آن ها را تک تک با جزییات توضیح می دهیم.

پیش پردازش داده

برای پردازش متن خبر مراحل زیر را به ترتیب انجام دادیم:

1) جایگزین کردن علائم نگارشی، که با دستور string.punctuation از کتابخانه string به دست می آیند، با white space
2) حذف white space های اضافی از متن خبر

3) lowercase کردن متن خبر
4) حذف اعداد از متن خبر
5) Tokenize کردن متن خبر (شکاندن به کلمات و نگه داشتن آن ها در یک لیست که با nltk انجام دادیم).
6) حذف stop word ها که تعریف آن ها پیشتر گفته شد (برای تشخیص stop word ها از کتابخانه nltk استفاده کردیم و لیستی از stop word ها در زبان انگلیسی را به وجود آوردیم تا کلماتی از متن خبر که در این لیست قرار میگیرند را حذف کنیم).
7) در گام آخر به ازای تمام کلمات باقی مانده از متن که فیلتر های گذشته را پشت سر گذاشته اند، عملیات stemming را انجام می دهیم تا هر کلمه با ریشه آن جایگزین شود و از تکرار کلمات هم ریشه جلوگیری کنیم. در رابطه با stemming و lemmatization در قسمت سوالات توضیح می دهیم.

پس از پیش پردازش متن خبر ها، یادگیری و تهیه لغت نامه را شروع می کنیم.

برای هر دسته، 80 درصد از خبر های آن را برای یادگیری و 20 درصد مابقی را برای ارزیابی مدل نگه می داریم.

برای مشخص کردن دسته یک خبر باید به ازای هر دسته مورد مطالعه، احتمال اون دسته به شرط متن خبر را حساب کنیم. متن خبر شامل لیستی از کلمات می باشد که فرض می کنیم احتمال وجود هر دو کلمه در یک دسته بندی مستقل از یکدیگر می باشند.
پس برای حساب کردن احتمال مربوط به یک دسته بندی خاص بودن متن خبر داریم:

$p(C_i|text)$ برای هر text هزینه ازای هر دسته

را حساب می‌کنیم.

$$p(C_i|text) = \frac{p(text|C_i)\,p(C_i)}{p(text)}$$

$$text = \text{list of words} = [w_1, w_2, \dots, w_i, \dots]$$

$$p(C_i|text) = \frac{p([w_1, w_2, \dots, w_i, \dots]|C_i)\,p(C_i)}{p(text)}$$

$$p(w_1, w_2, \dots, w_i, \dots|C_i) = p(w_1|C_i)\,p(w_2|C_i)\dots$$

↓

فرض می‌کنیم احتمال وجود هر واژه یک دسته‌بندی مستقل از واژه دیگر است.

همانطور که مشاهده می شود، طبق رابطه ی بالا برای محاسبه احتمال مربوط به دسته ci بودن یک متن خبر، با فرض استقلال هر دو واژه متن به شرط دسته ci بودن، باید به ازای هر واژه متن خبر، احتمال وجود آن در دسته بندی ci را به دست آورده و حاصل ضرب احتمالات واژه ها به شرط ci بودن را به دست آورده و حاصل را ضرب در احتمال ci بودن یک خبر و سپس تقسیم بر احتمال متن بکنیم.

منظورمان از استقلال واژه ها به شرط ci بودن این است که در صورتی که بدانیم یک متن ci است، احتمال وجود واژه w1 در آن مستقل از احتمال وجود w2 در آن است که البته نمی توانیم ادعای استقلال کامل بکنیم.

در نتیجه برای هر text، به ازای هر دسته ci احتمال بالا را مقایسه کرده و دسته با ماکسیمم احتمال را انتخاب می کنیم (توجه شود که چون p(text) در مخرج همه کسر های محاسبه بالا می آید و مقدار ثابتی داشته و به نوع دسته بستگی ندارد، از محاسبه آن صرف نظر می کنیم).

بدین منظور و برای به دست آوردن p(ci|text) باید برای هر کلمه آن، احتمال وجود آن کلمه به شرط عضو دسته ci بودن را بتوانیم به دست بیاوریم. همچنین احتمال ci بودن یک خبر را نیز باید بتوانیم به دست بیاوریم.

احتمال وجود یک کلمه به شرط آنکه بدانیم دسته بندی c می باشد، برابر با تعداد تکرار آن کلمه در دسته بندی c تقسیم بر تعداد کل کلمات (جمع تکرار های همه کلمات) دسته بندی c می باشد.
احتمال c بودن یک خبر نیز برابر با تعداد خبر های عضو دسته بندی c بودن داده آزمایشی تقسیم بر تعداد کل خبرهاست.

مجموعه خبر هایی که روی آن ها یادگیری را انجام می دهیم، همانطور که گفته شد 80 درصد از خبر های فایل data.csv می باشد که برای تهیه آن از هر دسته بندی، 80 درصد از خبر های آن را انتخاب کرده و در یک لیست نگه می داریم. 20 درصد مابقی را نیز برای ارزیابی مدل، بعدا با قاعده بیزین به ازای هر خبر به ازای دسته بندی ها احتمال گفته شده را حساب کرده و ماکسیسم را به عنوان دسته پیش بینی شده بیان می کنیم.

در مرحله اول (یادگیری)، تعداد کلمات هر دسته را ذخیره می کنیم تا در ارزیابی استفاده کنیم.
در مرحله دوم (ارزیابی)، به ازای هر متن خبر، احتمال p(ci|text) را برای هر دسته حساب می کنیم که روش گفته شده پیشتر ذکر شد.
سپس برای هر دسته خبر ضرایب precision ، recall را حساب کرده و accuracy کل را نیز در پایان حساب می کنیم.

recall برای یک دسته، تعداد خبر هایی از آن دسته می باشد که ما درست پیش بینی کرده ایم؛ تقسیم بر تعداد کل خبر های دسته.
precision برای یک دسته، تعداد خبر هایی از آن دسته می باشد که ما درست پیش بینی کرده ایم؛ تقسیم بر کل تعداد خبر هایی که ما به آن دسته نسبت داده ایم (که ممکن است شامل تعدادی خبر باشد که مربوط به دسته ای دیگرند ولی ما با استفاده از مدل خود آن ها را به اشتباه به دسته ذکر شده نسبت داده ایم)
accuracy نیز برابر با تعداد کل خبرهاییست که دسته بندی آن ها را درست تشخیص داده ایم؛ تقسیم بر تعداد کل خبر های دسته ارزیابی.

جدول های مربوط به این 3 معیار را به ازای هر فاز در شکل زیر مشاهده می کنید:

فاز اول (بررسی دسته های business و travel)

```
========== RESTART: C
Recalls : {'BUSINESS': 0.9247191011235955, 'TRAVEL': 0.8426966292134831}
Precisions : {'BUSINESS': 0.8546209761163032, 'TRAVEL': 0.9179926560587516}
Accuracy : 0.8837078651685393
```

فاز دوم (بررسی دسته های business و travel و style-beauty)

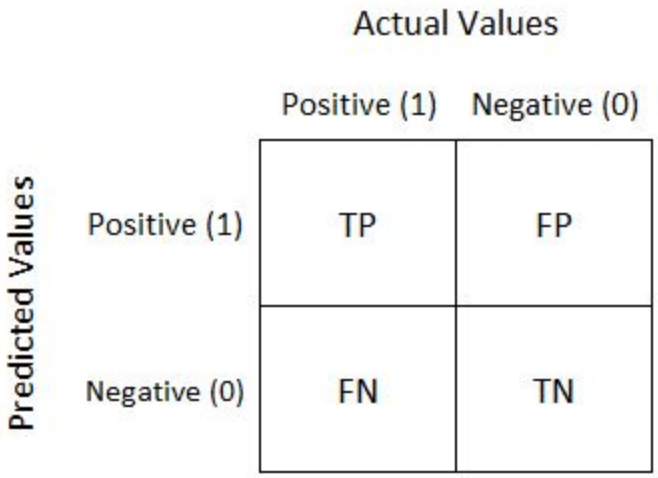
```
========== RESTART: C:\Users\hanıye\OneDrıve\Desktop\AICA3\
Recalls : {'BUSINESS': 0.9039325842696629, 'TRAVEL': 0.8404494382022472, 'STYLE & BEAUTY': 0.85}
Precisions : {'BUSINESS': 0.8341109383100052, 'TRAVEL': 0.8348214285714286, 'STYLE & BEAUTY': 0.9345274861025324}
Accuracy : 0.8647940074906367
```

توجه شود که در فاز یادگیری، از ستون های headline ، author و description برای تهیه لغت نامه و مدل سازی استفاده کرده ایم تا دقت بالاتر رود.

Confusion Matrix

Confusion matrix یک ماتریس برای ارزیابی مدلی که برای پردازش متن به دست آوردیم می باشد و بر روی داده های ارزیابی که دسته آن ها را می دانیم به دست می آید. در این ماتریس به ازای خبر های هر دسته مورد مطالعه (ci)، آن ها را به دسته هایی که نسبت داده شده اند تقسیم بندی می کنیم.

به عنوان مثال اگر دو دسته x و y را مطالعه می کنیم، برای خبر های دسته x، آنها را به دو دسته خبرهایی که به خود x نسبت داده شده اند و خبر هایی که به اشتباه به y نسبت داده شده اند تقسیم بندی می کنیم. همین عمل درباره دسته y نیز صادق است و در ماتریس به ازای هر دسته مورد مطالعه این اطلاعات را نگه می داریم.
در مثال دو دسته ای این ماتریس داریم:



در شکل بالا اگر دو دسته Positive و Negative را داشته باشیم،
برای دسته Positive،
TP تعداد خبر هایی از دسته Positive می باشد که مدل ما به درستی پیش بینی کرده.
FN تعداد خبر هایی از دسته Positive می باشد که مدل ما به اشتباه Negative پیش بینی کرده.

و برای دسته Negative،
TN تعداد خبر هایی از دسته Negative می باشد که مدل ما به درستی آن ها را Negative پیش بینی کرده.
FP تعداد خبر هایی از دسته Negative می باشد که مدل ما به اشتباه Positive پیش بینی کرده.

با استفاده از این داده ها می توان precision ، recall هر دسته و accuracy کل را به دست آورد.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

ضریب دیگری به اسم F-measure نیز قابل تعریف است که برای نشان دادن ترکیبی از دو معیار recall و precision به کار می رود و به مقدار کوچک recall یا precision نزدیک است ولی چون در صورت پروژه خواسته نشده بود آن را حساب نمی کنیم.

$$F - measure = \frac{2*Recall*Precision}{Recall + Precision}$$

Confusion matrix مربوط به فاز دوم ارزیابی داده ها که شامل 3 دسته مورد مطالعه بود مطابق زیر است:

confusion_matrix : {'BUSINESS': {'BUSINESS': 1621, 'TRAVEL': 109, 'STYLE & BEAUTY': 50, 'Total': 1780}, 'TRAVEL': {'TRAVEL': 1420, 'BUSINESS': 275, 'STYLE & BEAUTY': 85, 'Total': 1780}, 'STYLE & BEAUTY': {'STYLE & BEAUTY': 1530, 'TRAVEL': 130, 'BUSINESS': 120, 'Total': 1780}}

همانطور که پیش تر توضیح داده شد، در این ماتریس به ازای خبر های هر دسته آن ها را به دسته هایی که پیش بینی شده اند تقسیم کرده و چاپ کرده ایم.

نمونه برداری

در ابتدا خبرهای فایل data.csv را یکی یکی خوانده و در دسته مربوطه قرار دادیم.
پس از پردازش و یادگیری داده ها در مرحله ارزیابی متوجه شدیم که تفاوت precision های دسته های مختلف زیاد است.

```
========== RESTART: C:\Users\hanıye\OneDrıve\Desktop\AICA3
Recalls : {'BUSINESS': 0.7455565949485501, 'TRAVEL': 0.8460674157303371, 'STYLE & BEAUTY': 0.8578008059873344}
Precisions : {'BUSINESS': 0.718018018018018, 'TRAVEL': 0.814935064935065, 'STYLE & BEAUTY': 0.9152334152334153}
Accuracy : 0.8270824247710423
```

در شکل بالا مشاهده می شود که تفاوت precision های دسته ها زیاد است (مثلا دسته business و دسته style-beauty تفاوت زیادی دارند).

علت میتواند این باشد که در مرحله یادگیری تعداد خبر های با دسته بندی business بسیار کمتر از خبر های دو دسته دیگر است و ازاین رو اطلاعات زیادی راجع به کلمات به کار رفته در این دسته در مرحله یادگیری نخواهیم آموخت و تعداد خبرهای درستی که بعدا به این دسته نسبت می دهیم (در مرحله ارزیابی) از واقعیت کمترند و ضرایب precision و recall بالایی به دست نخواهد آمد.

برای حل این مشکل، باید تعداد خبر های مربوط به دسته اقلیت را بیشتر کنیم تا این عدم تعادل در تعداد خبر های بین دسته های مختلف در مرحله یادگیری جبران شود و در ارزیابی پیش بینی دقیق تری حاصل شود. این روش oversampling برای متعادل کردن داده ها نام دارد و برای پیاده سازی آن روش های مختلفی از جمله random oversampling و استفاده از روش smote وجود دارد که کتابخانه های آن ها در پایتون موجود می باشند.

برای سادگی کار و پرهیز از استفاده از کتابخانه، برای افزایش تعداد خبر های دسته های اقلیت، برای دو دسته style-beauty و business که تعداد خبر کمتری نسبت به دسته travel دارند، به تعداد خبری که کمتر دارند، از خبر هایشان به صورت رندم انتخاب کرده و به مجموعشان کپی و اضافه می کنیم. مثلا اگر دسته business را در نظر بگیریم به تعداد (اعضای travel منهای اعضای business) از بین خبر های business رندوم انتخاب کرده و به آن اضافه می کنیم تا تعداد اعضای دسته ها و پراکندگی واژه های دسته ها متعادل تر شوند.

پس از نمونه برداری، تفاوت recall ها و precision های دسته های مختلف کمتر شده و همانطور که در عکس ها قبلا نشان دادیم، تفاوت ضرایب دسته ها حداکثر 10-11 درصد می باشد که قابل چشم پوشی و مطلوب است.

در نهایت داده های test.csv را با مدل خود ارزیابی کرده و در فایل output.csv نتایج را نشان دادیم.

سوالات

1) بررسی stemming و lemmatization برای به دست آوردن ریشه کلمات متن؟

هر دو روش برای تبدیل واژه ها به یک ریشه مشترک و حذف قسمت های اضافی کلمات مانند s آخر کلمات جمع هستند.stemming فرایندی است که کلمات را به شکل ریشه خود کاهش می دهد و آن ها را به یک فرم مشترک می برند حتی اگر ریشه معنی لغت نامه نداشته باشد. به عنوان مثال:
beautiful و beautifully به beauti مشتق می شوند که در فرهنگ لغت انگلیسی معنی ندارد. این روش معمولا برای رسیدن به این هدف انتهای کلمات را می برد.

lemmatization فرایندی است برای کاهش کلمات به ریشه زبان یا فرهنگ لغت آنها و معنی کلمه در جمله را در نظر می گیرد. به عنوان مثال: beautiful و beautifully به beautiful مشتق می شوند که در لغت نامه معنی دارد. این روش با استفاده از واژگان یک لغت نامه و با تحلیل مورفولوژیک کلمات آن ها را به ریشه واقعی و معنا دار خود مشتق می کند.

مثلا کلمه better به good توسط lemmatization مشتق می شود؛ در حالیکه stemming آن را به همان better مشتق می کند.
کلمه walk در هر دو روش به walk مشتق می شود و کلمه meeting بنا به کاربرد آن در متن توسط lemmatization به meet یا خود همان meeting مشتق می شود حال آنکه stemming آن را در هر شرایط به meet مشتق می کند.

در بررسی محتوای اخبار، stemming سریعتر عمل می کند ولی ممکن است ریشه واقعی کلمات را نسازد (فقط بعضی پسوند ها و پیشوند ها را حذف می کند) وهمانطور که توضیح داده شد lemmatization دقیق تر عمل می کند و ریشه واقعی کلمات را می سازد؛ پس در برای بالا بردن دقت کار خود از lemmatization استفاده کردیم (دقت را به زمان ترجیح دادیم).

2) TF * IDF چیست؟
الگوریتم TF * IDF برای ارزش گذاری (وزن کردن) یک کلمه کلیدی در هر محتوا و تعیین اهمیت آن کلمه کلیدی بر اساس تعداد دفعات مشاهده شده در سند استفاده می شود. هم چنین این معیار بررسی می کند که کلمه مذکور در کل مجموعه نیز چقدر ارزشمند است.

برای یک کلمه t در سند d، ارزش کلمه در سند با wt,d مشخص می شود که خود برابر است با:

$$Wt,d = TFt,d * \log (N/DFt)$$

TFt,d تعداد دفعاتی است که کلمه t در سند d مشاهده شده است.
در حقیقت tf برای یک کلمه، فرکانس آن کلمه (تعداد تکرار کلمه) در یک متن را نشان می دهد.
مثلا اگر یک سند 100 کلمه ای 30 بار کلمه computer را داشته باشد، tf برای کلمه computer برابر است با 0.3.

از طرفی idf اهمیت کلمه در کل متن را نشان می دهد.
شاخص IDF همان log(DF) است که به عنوان مثال اگر فرض کنیم در مجموعه 1000 سند داریم که 30 تا از آن ها کلمه computer را دارند، idf کلمه computer برابر با log(1000/30) می شود.

در نتیجه:

W computer = (TF*IDF) computer = 0.3 * log(1000/30) = 0.3 * 1.52 = 0.456

TF * IDF را برای کلمات خود اجرا کرده و وزن آنها را بدست می آوریم. هرچه مقدار وزن عددی بالاتر باشد ، این کلمه کمیاب تر است و هرچه وزن کوچکتر باشد ، این کلمه رایج تر است.

اگر بخواهیم از این شاخص در روابط بیزین استفاده کنیم، به جای شاخص تعداد کلمه در متن، باید وزن tf-idf آن کلمه در متن را در به کار میبریم:

Word t count in Class c:

جمع همه وزن های tf-idf کلمه t به ازای هر متنی (سندی) که متعلق به دسته c باشد.

All words belong to Class c count:

جمع tf-idf همه ی کلمات متعلق به دسته c (برای هر کلمه باید tf-idf آن به ازای هر متن متعلق به دسته c را به دست آورده و جمع کنیم)

3)

همانطور که در تعریف precision گفتیم، اگر precision برای یک دسته بالا باشد، بدین معناست که

از میان خبر هایی که ما آن ها را به آن دسته نسبت داده ایم، جمع زیادی واقعا مربوط به دسته c هستند و پیش بینی غلط درباره این کلاس کمتر کرده ایم. بالا بودن این شاخص به تنهایی کافی نیست.

چرا که ممکن است ما در پیش بینی هایمان راجع به یک دسته خطای کمی داشته باشیم اما پیش بینی مان کامل نبوده باشد (یعنی تعداد کل پیش بینی هایی که به آن دسته خاص نسبت داده ایم کمتر از تعداد واقعی خبر های مربوط به آن دسته باشد). مثلا فرض کنید که از دسته A، تعداد 20 نمونه داریم و مدل ما ذر ارزیابی خود 10 خبر را به دسته A نسبت داده که 9 تای آن ها واقعا مربوط به دسته A هستند.

در این جا precision مقدار 0.9 دارد که عدد خوبیست اما recall مقدار 9/20 دارد که نشانگر ضعیف عمل کردن مدل در پیش بینی تعداد درست خبرهای متعلق به دسته A می باشد.

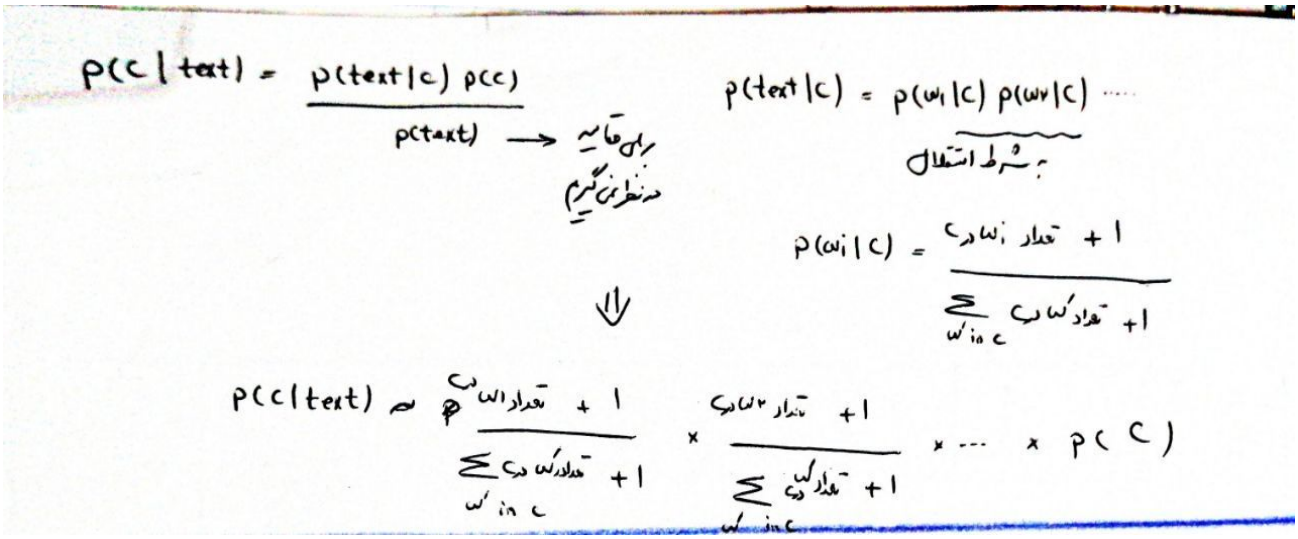
پس علاوه بر شاخص precision، باید recall را هم مدنظر داشته باشیم.

به عنوان مثال یک ابزار تشخیص موقعیت عکس را در نظر بگیرید که بر اساس عکس که دریافت می کند باید تشخیص دهد که عکس به جنگل، ساحل یا کوهستان مربوط است.

فرض کنیم ابزار در هنگام یادگیری بگونه ای آموزش دیده که اگر در عکس درخت دید آن را با احتمال خوبی به دسته جنگل نسبت می دهد.

حال اگر تعدادی عکس در اختیار ماشین بگذاریم که در تعدادی از آنها که مربوط به جنگل هستند عکس درخت باشد (10 عدد) ولی در تعدادی از آنها که اتفاقا مربوط به جنگل هستند عکس درخت نباشد (10 عدد)، ماشین ما ممکن است تعداد 12 عکس را به جنگل نسبت دهد که اتفاقا 10 تای آنها عکس درخت دارند. پس precision در اینجا 10/12 می شود و عدد خوبی است. حال آنکه تقریبا همه عکس های جنگل که درخت ندارند را miss کرده ایم و recall برابر با 10/20 می شود که خوب نیست و این ضعف ما در مدل کردن هنگام یادگیری را نشان می دهد.

(4



همانطور که قبلا توضیح داده شد، برا محاسبه احتمال وجود کلمه به شرط یک دسته بندی،

به به ازای هر کلمه موجود در دسته بندی یک آفست برابر با 1 می دهیم. یعنی تعداد یک کلمه در دسته بندی را به علاوه یک می کنیم در هنگام محاسبه تا در صورتی که هنگام تست یک کلمه مشاهده شده پیشتر در یک دسته بندی نبود، احتمال وجود آن کلمه در دسته بندی 0 نشود که منجر به 0 شدن احتمال وجود متن خبر حاوی آن کلمه در آن دسته بندی می شود که غلط است.

پس کلمه travel بنا به مقدار p(ci) و تعداد کلمات موجود در ci و دیگر مواردی که در فرمول بالا نشان داده شده اند، احتمال وجود در ci را دارد (هر چند اگر کم باشد) و باید به ازای تمام دسته ها احتمال را حساب کرده و ماکسیمم بگیریم.