بسمه تعالی

گزارش تمرین عملی ۱ درس آمار واحتمال مهندسی

عنوان: تشخیص ایمیل های سالم و هرزنامه با مدل بیز

نام و نام خانوادگی: سینا مظاهری شماره دانشجویی : ۹۸۱۷۱۱۵۹

تئوری پیاده سازی

برای پیاده سازی خواسته سوال ابتدا مفهوم مکانیزم محاسبه احتمال را برای برنامه نوشته شده توضیح می دهیم. در ابتدا ما فضای نمونه ای را برای اینکه یک ایمیل توصیف شود، بیان می کنیم. ما در حل این سوال فرض کردیم که یک ایمیل، به شکل برداری متشکل از n کلمه می باشد. حال اگر طول ایمیل کوچکتر از این مقدار باشد مقادیر متناظر در جایگاه آن کلمه را به شکل null (" ") تعریف می کنیم تا همگی ایمیل ها دارای یک اندازه شوند. حال اگر ایمیلی، طولی بیشتر از مقدار n داشته باشد، جایگاه کلمات بزرگتر از n آن، از بین می رود. نکته ی دیگری که در حل این سوال در نظر گرفته شده آن است که جایگاه کلمات از یک تا n، با رویداد یک یا چند کارکتر سفید یعنی کاراکتر گریز به خط بعد (\n) ، کاراکتر گریز tab (\t) ، کاراکتر blank (\b) یا همان فاصله و همچنین کاراکتر پوچ یا null، از یکدیگر جدا و تمیز داده شده اند. به منظور رفع ابهام با توجه به آن چه که در بالا گفته شد ما ابتدا کلمات ایمیل را بر اساس کاراکتر های سفید جدا می کنیم و اگر با طول پیش فرض ایمیل یعنی n برابر نبود جایگاه های خالی را با null پر می کنیم تا هم سایز شوند. حال از آن جا که کلاس های طبقه بندی شونده ما از دو نوع ایمیل عادی (H) و هرزنامه (S) می باشند، برای جایگاه هر کلمه در هر دو نوع ایمیل، دو فضای نمونه ای مجزا در نظر گرفته شده که با اندیس H یا S از یکدیگر متمایز شده اند. به طور مثال، مجموعه تمامی کلمات ممکن که در جایگاه اول ایمیل های عادی می توانند بیایند به صورت Ω_{1H} می باشد. همچنین در یک مثال دیگر، مجموعه کلماتی که در جایگاه چهارم ایمیل های هرز نامه می آیند به صورت Ω_{4S} می باشد. شکل کامل تر آن به شرح زیر می باشد. لازم به ذکر است که این مجموعه ها از روی ایمیل های تمرینی ساخته خواهند شد.

$$\Omega_{iX} = \{set\ of\ all\ training\ words\ of\ X's\ , which\ they\ come\ at\ the\ i\ th\ position\}$$

حال که توانستیم مجموعه لغات هر جایگاه ایمیل های هرزنامه و عادی را پیدا کنیم، برای آن ها یک تابع توزیع احتمال در نظر می گیریم. تابعی که ما برای حل مسئله در نظر گرفتیم تابع توزیع احتمال یکنواخت گسسته می باشد که به تصادفی ترین شکل ممکن احتمال را توزیع می کند بدین صورت که تمامی اعضای فضای نمونه ای شانس وقوع یکسانی دارند. با توجه به این موضوع، کافیست به ازای هر عضو موجود در فضای نمونه ای یک احتمال تعریف شود. از آن جا که ۳۰۰ ایمیل تمرینی، چه برای هرزنامه و چه برای عادی وجود دارد احتمال وقوع هر عضو در فضای نمونه ای Ω_{iX} به شکل زیر خواهد بود:

$$P(\omega_i \mid X) = \frac{number\ of\ \omega\ occurrences\ in\ i\ th\ position\ of\ X's\ emails}{300}$$

در قسمت پیاده سازی کد در مورد این که با چه داده ساختاری آن را پیاده سازی می کنیم، توضیح خواهیم داد. حال با خواندن ایمیل ها و ذخیره مقادیر احتمال مشاهده شده به ازای هر کلمه، مکانیزم دسته بندی و یا تشخیص یک ایمیل را شرح خواهیم داد. برای اینکه تشخیص دهیم یک ایمیل به کدام دسته تعلق دارد باید ابتدا احتمال های عادی بودن به شرط ایمیل بودن یعنی ($P(H|E)$) و همچنین احتمال هرزنامه بودن به شرط ایمیل بودن یعنی ($P(S|E)$) را به طور جداگانه با استفاده از قاعده بیز محاسبه نموده و سپس برچسب آن دسته را بدهیم که احتمال بیشتری دارد. به همین منظور قاعده بیز را یاد آوری می کنیم.

$$P(S|E) = \frac{P(E|S)P(S)}{P(E)}$$

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

در این دو عبارت بالا مقدار $P(E)$ همان جمع دو صورت کسر بالا، یعنی حاصل زیر می باشد.

$$P(E) = P(E|S)P(S) + P(E|H)P(H)$$

حاصل عبارات $P(S)$ و $P(H)$ را به شکل زیر حساب می کنیم از آن جا که هر کدام ۳۰۰ ایمیل در دسته تمرینی خود دارند، بنابراین احتمال آن که ایمیلی هرزنامه یا سالم با هم برابر و مقدار $\frac{1}{2}$ را دارند زیرا:

$$P(H) = P(S) = \frac{300}{300+300} = \frac{300}{600} = \frac{1}{2}$$

حال برای آن که احتمال های $P(E|S)$ و $P(E|H)$ را محاسبه کنیم نیازمندیم تا فرض مستقل شرطی بودن آمدن کلمات در ایمیل را رعایت کنیم. در اینجا به این سوال پاسخ می دهیم که چرا فرض مستقل بودن کلمات در دنیای واقعی اشتباه است. در پاسخ به این سوال باید گفت که فرض مستقل بودن رویداد کلمات در قسمت های مختلف یک عبارت (جمله) اشتباه است زیرا دنیای واقعی برای هر کلمه نقش معینی تعریف شده است به طور مثال در زبان فارسی کلمات نقش نهاد و متمم و مفعول و مسند دارند که ترتیب آمدن آن ها به شکل خاصی می باشد یعنی در اغلب جملات فعل بعد از نهاد می آید. از همین واژه ی اغلب می توان نتیجه گرفت که احتمال آمدن چنین جملاتی بیشتر است در حالی که در فرض مستقل بودن احتمال آن که ابتدا نهاد بیاید و سپس فعل یا بلعکس، یکسان توصیف می شد. به صورت مسئله خود باز می گردیم. بنابه آن چه گفته شد و قانون استقلال شرطی خواهیم داشت:

$$P(E|S) = P\big((\omega_1,\ \omega_2,\ \omega_3, \dots,\ \omega_n)\big|S\big) = \prod_{i=1}^{n} P(\omega_i|S)$$

$$P(E|H) = P\big((\omega_1,\ \omega_2,\ \omega_3, \dots,\ \omega_n)\big|H\big) = \prod_{i=1}^{n} P(\omega_i|H)$$

پیاده سازی کد

مناسب ترین داده ساختار برای ذخیره احتمال کلمات موجود در یک جایگاه دیکشنری یا همان داده ساختار **dict()** پایتون می باشد حال برای هر جایگاه یک دو تایی **(tuple)** تعریف می کنیم که المان اول آن به دیکشنری احتمال ایمیل عادی و دومی به هرزنامه اشاره دارد. حال لیستی از این **n** تا **tuple** دوتایی تعریف می کنیم که هر کدام آن مربوط به **tuple** دیکشنری های احتمالاتی لغات جایگاه مورد نظر باشد. این مقادیر با تابع زیر مقدار دهی اولیه می شوند که در ورودی طول پیش فرض ایمیل ها یعنی **n** را می گیرد.

```
def initialize_the_table(size):    # size is the default length of the email
    the_table = list()
    for i in range(size):
        the_table.append((dict(), dict()))
    return the_table
```

در قسمت بعدی تابعی که سبب ایجاد مدل احتمالاتی می شود را شرح خواهیم داد. در این قسمت تابع مورد نظر مسیر دایرکتوری فایل های برچسب دار، جدول احتمالاتی ، شماره دسته کلاس (۰ برای عادی و ۱ برای هرز نامه) و همچنین تعداد ایمیل های موجود در آن را گرفته و سپس جدول احتمالاتی ما را می سازد که فاقد هر گونه کاراکتر سفید می باشد به شرط آن که طول آن بزرگتر یا مساوی **n** باشد. در غیر اینصورت این اختلاف فاصله با **null** پر می شود.

```
def get_the_word_probability_table_by_emails(path_address, class_num, table, number_of_email):
    for i in range(1, number_of_email + 1):
        email = open(path_address + '(' + str(i) + ')' + ".txt", 'r', encoding='utf-8-sig')
        email_content = email.read()
        list_of_words = list(filter(lambda x: len(x) != 0, re.split("\\s+", email_content)))  # filter the white
        # characters
        iteration_number = min(len(table), len(list_of_words))  # choose the minimum length between default and n
        for j in range(iteration_number):  # compute probabilities
            feature_dic = table[j][class_num]
            word = list_of_words[j]
            feature_dic[word] = feature_dic.get(word, 0) + (1 / number_of_email)
        for j in range(len(table) - iteration_number):  # fill the rest of the length with blank
            feature_dic = table[j + iteration_number][class_num]
            feature_dic[''] = feature_dic.get('', 0) + (1 / number_of_email)
        email.close()
    return table
```

بعد از انجام دو مرحله بالا نوبت به پیش بینی و طبقه بندی ایمیل های ناشناخته می گردد. در ابتدای تابع، نوع کلاس ایمیلی که آمده در **class_name** ذخیره می گردد. سپس با انجام یک حلقه برروی کلمات ایمیل احتمالات $\prod_{i=1}^{n} P(\omega_i|H)$ و همچنین $\prod_{i=1}^{n} P(\omega_i|S)$ را محاسبه می کنیم. نکته ی مهمی که در اینجا وجود دارد آن است که اگر کلمه ای در ایمیل ناشناخته وجود داشت که نظیر آن در ایمیل های تمرینی وجود نداشت، ما آن را در نظر نمی گیریم و احتمالات را با توجه به کلمات سایر جایگاه های دیگر محاسبه می کنیم.

در نهایت نیز، احتمال $P(E)$ را با توجه به داشتن چهار احتمال $P(H), P(S)$ و همچنین $P(E|H)$, $P(E|S)$ حساب می کنیم و با استفاده از قاعده بیز احتمال های $P(H|E)$ و $P(S|E)$ را می یابیم و تصمیم گیری را بر اساس بزرگی هر کدام انجام می دهیم. بدین منظور دو پوشه **spam prediction** و همچنین **ham prediction** تعبیه شده که نتیجه پیش بینی ما می باشد همچنین **class_name** برروی اسم فایل متنی قرار می گیرد تا بتوانیم دقت مدل را اندازه گیری نماییم.

```
def classify_the_emails(source_path, number_of_emails, table, total_training_emails, number_of_hamtraining_emails,
                        number_of_spamtraining_emails):
    if "ham" in source_path:
        class_name = "ham"
    else:
        class_name = "spam"
    n = len(table)
    for i in range(1, number_of_emails + 1):
        target_path = '(' + str(i) + ')' + ".txt"
        target_email = open(source_path + target_path, 'r', encoding="utf-8-sig")
        target_content = list(filter(lambda x: len(x) != 0, re.split("\\s+", target_email.read())))
        iteration_number = min(n, len(target_content))
        probability_of_being_E_respect_to_condition_H = 1  # P(E|H)
        probability_of_being_E_respect_to_condition_S = 1  # P(E|S)
        for j in range(iteration_number):
            word = target_content[j]
            feature_dic_ham = table[j][0]
            feature_dic_spam = table[j][1]
            p_0 = feature_dic_ham.get(word, 0)
            p_1 = feature_dic_spam.get(word, 0)
            if p_0 == 0 or p_1 == 0:
                continue
            probability_of_being_E_respect_to_condition_H = p_0 * probability_of_being_E_respect_to_condition_H
            probability_of_being_E_respect_to_condition_S = p_1 * probability_of_being_E_respect_to_condition_S
        target_email.close()
        evidence_probability = (
                                    probability_of_being_E_respect_to_condition_H * number_of_hamtraining_emails / total_training_emails) + \
                               (
                                    probability_of_being_E_respect_to_condition_S * number_of_spamtraining_emails / total_training_emails)
        # compute evidence probability = P(E) = P(E|H)P(H) + P(E|S)P(S)
        probability_of_being_H_respect_to_condition_E = ((number_of_hamtraining_emails / total_training_emails) *
                                                         probability_of_being_E_respect_to_condition_H) / evidence_probability
        # compute P(H|E) = P(E|H)P(H) / P(E)
        probability_of_being_S_respect_to_condition_E = 1 - probability_of_being_H_respect_to_condition_E  # compute P(S|E) = 1 - P(H|E)
```

در نهایت نیز با انجام تمامی فرآیند های بالا، فایل ها دسته بندی شده و دقت مدل به شرح زیر می باشد:

دقت تشخیص:

$$accuracy\ of\ correctness\ \ classifyng = \frac{138 + 126}{200 + 200} = \frac{264}{400} = 66\%$$