بسمه تعالی

Naïve Bayesian spam filtreing

متین داغیانی

دانشگاه صنعتی شریف

بهار ۱۴۰۰

فهرست

مقدمه

در این پروژه قصد داریم تا به کمک قانون بیز، مکانیزمی را طراحی و پیاده‌سازی کنیم که بتواند هرزنامه‌ها را از ایمیل‌های سالم تشخیص و جداسازی کند. برای این کار از دو Data set ، هر کدام شامل ۳۰۰ ایمیل اسپم/غیراسپم برای یادگیری (train) و از دو Data set دیگر هر کدام شامل ۲۰۰ ایمیل برای تست استفاده شده است. در ادامه به بررسی مدل و نحوه پیاده‌سازی می‌پردازیم. i

بخش نظری

فرضیات اولیه

۱- ایمیل E متشکل از واژگان w1 ، w2 ، ... w_n , را در نظر بگیرید. اگر احتمال وقوع رخداد E را با P(E) نمایش دهیم، می‌توان فرض کرد که :

$$P(E) = \prod_{i=1}^{n} P(w_i)$$

که در آن $P(w_i)$ احتمال ظاهر شدن کلمه w_i در ایمیل E است.

- **چرا این فرض در واقعیت درست نیست؟**

توجه داشته باشید که احتمال وقوع کلمات از یکدیگر مستقل نیستند. به عنوان مثال احتمال وجود کلمه «*خوشبو*» در ایمیلی که در مورد *روش های نوین آبیاری گل‌ها* است و در آن واژگانی مثل *گل، گلدان ، باغ* بسیار ظاهر شده‌اند، بسیار بیشتر از ایمیلی است که در مورد *ذخیره داده‌ها در بسترهای ابری* بوده و از واژگان کاملا متفاوتی تشکیل شده‌اند.

به طور دقیق‌تر وجود یا عدم وجود برخی کلمات می‌توانند بر احتمال ظاهر شدن برخی واژگان دیگر تاثیر قابل توجهی داشته باشند.

۲- در این مدل‌سازی با توجه به این که مجموعه ایمیل‌های در دسترس برای یادگیری به طور برابر بین ایمیل‌های اسپم و غیر اسپم تقسیم شده‌اند، فرض شده است که احتمال هرز بودن (S) و غیرهرز بودن (H) یک ایمیل با هم برابر هستند. به عبارت دیگر:

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

محاسبات احتمالات

برای تشخیص اسپم بودن یک ایمیل قصد داریم تا به کمک قانون بیز احتمالات $P(S|E)$ و $P(H|E)$ را محاسبه کرده و بایکدیگر مقایسه کنیم. اگر $P(S|E)$ از $P(H|E)$ بیشتر بود، ایمیل هرزنامه و در غیر این صورت عادی تلقی می‌شود. در ادامه به مراحل طی شده برای محاسبه این دو احتمال می‌پردازیم.

۱- ابتدا نیاز است که $P(w|S)$ و را محاسبه کنیم. عبارت فوق بیانگر *احتمال وقوع کلمه* w *به شرط اسپم بودن ایمیل* است. به همین ترتیب $P(w|H)$ را نیز تعریف می‌کنیم.

برای محاسبه هر یک از روابط زیر استفاده می‌کنیم:

$$P(w|S) = \frac{word\ count\ in\ spam\ distribution}{total\ words\ in\ spam\ distribution}$$

$$P(w|H) = \frac{word\ count\ in\ ham\ distribution}{total\ words\ in\ ham\ distribution}$$

۲- در ادامه لازم است تا احتمالات $P(S|w)$ و $P(H|w)$ را محاسبه کنیم. $P(S|w)$/*احتمال اسپم بودن ایمیل به شرط وجود کلمه* w را نشان می‌دهد. به همین ترتیب $P(H|w)$ تعریف می‌شود.

برای محاسبه این احتمالات از قانون بیز استفاده می‌کنیم و حالات زیر را در نظر می‌گیریم:

- اگر **w** در میان *پربسامدترین واژگان ایمیل اسپم* [ii] بوده ولی در میان *پربسامدترین واژگان ایمیل عادی* نباشد :

$$P(S|w) = 0.999 \quad and \; P(H|w) = 0.001$$

- اگر **w** در میان *پربسامدترین واژگان ایمیل عادی* بوده ولی در میان *پربسامدترین واژگان ایمیل اسپم* نباشد :

$$P(S|w) = 0.001 \quad and \; P(H|w) = 0.999$$

- اگر **w** در میان هیچ کدام از پر بسامدترین واژگان اسپم و غیر اسپم نباشد، به این معناست که این واژه چندان رایج نیست. لذا :

$$P(S|w) \; = \; P(H|w) \; = \; 0.5$$

- اگر ***w*** در هر دو پربسامدترین واژگان وجود داشت، داریم:

$$P(S|w) \; = \; \frac{P(w|S)P(S)}{P(w|S)P(S) \; + \; P(w|H)P(H)} \; = \; \frac{P(w|S)}{P(w|S) \; + \; P(w|H)}$$

در ادامه تابع پیاده‌سازی شده برای محاسبه این احتمالات را مشاهده می‌کنید.

```python
def get_spam_if_word(word, ham_distribution: dict, spam_distribution):
    """
    P(S|w) = P(w|S) / P(w|S) + P(w|H)
    P(w|S) = word_spam_count/total_spam_count
    P(w|H) = word_ham_count/total_ham_count

    """

    hams_total = sum(ham_distribution.values())
    spams_total = sum(spam_distribution.values())
    spam_most_frequents = spam_distribution.keys()
    ham_most_frequents = ham_distribution.keys()
    if (word in spam_most_frequents) and (word not in ham_most_frequents):
        return 0.999
    elif (word in ham_most_frequents) and (word not in spam_most_frequents):
        return 0.001
    elif word not in (spam_most_frequents or ham_most_frequents):
        return 0.5
    else:
        word_if_spam = spam_distribution[word] / spams_total
        word_if_ham = (ham_distribution[word]) / hams_total
        return word_if_spam / float(word_if_spam + word_if_ham)
```

۳- در نهایت بار دیگر برای محاسبه احتمالات مورد نیاز، از قانون بیز کمک می‌گیریم. خواهیم داشت:

$$P(S|E) = \frac{P(E|S)P(S)}{P(E|S)P(S) + P(E|H)P(H)} = \frac{P(E|S)}{P(E|S) + P(E|H)}$$
$$= \frac{\prod_{i=1}^{n} P(w_i|S)}{\prod_{i=1}^{n} P(w_i|S) + \prod_{i=1}^{n} P(w_i|H)}$$
$$= \frac{\prod_{i=1}^{n} \frac{P(S|w_i)P(w_i)}{P(S)}}{\prod_{i=1}^{n} \frac{P(S|w_i)P(w_i)}{P(S)} + \prod_{i=1}^{n} \frac{P(H|w_i)P(w_i)}{P(H)}}$$
$$= \frac{\prod_{i=1}^{n} P(S|w_i) \prod_{i=1}^{n} P(w_i)}{\prod_{i=1}^{n} P(S|w_i) \prod_{i=1}^{n} P(w_i) + \prod_{i=1}^{n} P(H|w_i) \prod_{i=1}^{n} P(w_i)}$$
$$= \frac{\prod_{i=1}^{n} P(S|w_i)}{\prod_{i=1}^{n} P(S|w_i) + \prod_{i=1}^{n} P(H|w_i)}$$
$$= \frac{\prod_{i=1}^{n} P(S|w_i)}{\prod_{i=1}^{n} P(S|w_i) + \prod_{i=1}^{n} (1 - P(S|w_i))}$$

در ادامه تابع پیاده‌سازی شده برای تشخیص اسپم بودن ایمیل با توجه با محاسبات بالا را مشاهده می‌کنید.

توجه داشته باشید برای جلوگیری از به دست آمدن مقادیر بسیار کوچک که می‌توانند محاسبات کامپیوتر را دچار مشکل کنند، تنها کلماتی در نظر گرفته شده‌اند که احتمال اسپم بودن آن‌ها یا تا حد بسیار خوبی نزدیک به ۱ یه تاحد بسیار خوبی نزدیک به ۰ باشد. به عبارتی، تنها ۲۰۰ کلمه با بیشترین انحراف از احتمال ۰.۵ در نظر گرفته شده‌اند.

```python
def is_spam(email_path: str, ham_distribution, spam_distribution):
    """To prevent very small probabilities (so division by zero)
    , top 200 words with very high or very low probabilities are chosen"""
    email_words = get_words(email_path)
    top_words = {}
    for word in email_words:
        top_words[word] = abs(get_spam_if_word(word, ham_distribution, spam_distribution) - 0.5)
    top_words = sorted(top_words, key=top_words.get, reverse=True)
    spam_if_email = 1
    ham_if_email = 1

    for word in top_words[:200]:
        p = get_spam_if_word(word, ham_distribution, spam_distribution)
        spam_if_email *= p
        ham_if_email *= (1 - p)
    if spam_if_email > ham_if_email:
        return True
    return False
```

بخش پیاده‌سازی

آماده‌سازی داده‌ها

۱- ساخت توزیع واژگان اسپم و غیر اسپم:

برای محاسبه احتمالات لازم است تا کلمات پربسامد (پرتکرار) در ایمیل‌های اسپم و عادی را پیداکنیم. برای این‌کار تک‌تک واژگان به کار رفته در ایمیل‌های اسپم/عادی را بررسی کرده و تعداد آن‌ها را می‌شماریم. در نهایت یک دیکشنری شامل لغات یکتا به عنوان کلید و تعداد آن‌ها به عنوان مقدار ایجاد خواهیم کرد. در نهایت ۵۰۰ واژه پر بسامد را در نظر می‌گیریم.

۲- کلمات توقف:

کلمات توقف یا stop words واژگانی هستند که در هر نوشتار به صورت مکرر و معمول استفاده می‌شوند. این واژگان شامل افعال، علائم نگارشی، حروف اضافه و ... پرکاربرد هستند.

وجود این واژگان می‌توانند در نتایج بدست آمده تاثیر داشته باشند و یا سرعت پردازش را کاهش دهند. به همین خاطر لیستی از این واژگان در یک فایل در کنار پروژه قرار داده شده است و هنگام ساخت توزیع این واژگان در نظر گرفته نمی‌شوند.[iii]

```python
def get_words(path):
    with open(path, 'r', encoding='utf-8') as file:
        return [word for word in file.read().split() if word not in COMMON_WORDS]


def get_distribution(sample_path, k=500):

    distribution = collections.defaultdict(lambda: 1)
    files = os.listdir(sample_path)

    for f_name in files:
        list_of_words = get_words(sample_path + '/' + f_name)
        for word in list_of_words:
            distribution[word] += 1

    distribution = dict(sorted(distribution.items(), key=lambda x: x[1], reverse=True))

    return dict(list(distribution.items())[:k])
```

نتایج

پس از اجرای برنامه می‌توانید نتیجه تشخیص مکانیزم طراحی شده برای هر ایمیل از پوشه‌های اسپم و غیر اسپم را به صورت جدا گانه ملاحظه کنید. هم‌چنین در انتها نسبت تعداد پیام‌هایی که به درستی تشخیص داده‌شداند به تعداد کلا قابل مشاهده است.

این نتایج در فایل result.txt در پوشه پروژه پیوست شده است.

[i] در این گزارش واژگان اسپم و هرزنامه به یک معنا و غیراسپم و عادی به یک معنا درنظر گرفته شده اند

[ii] واژگانی که به صورت مکرر در ایمیل های اسپم مشاهده می شوند. پیاده سازی در ادامه آمده است

[iii] https://github.com/kharazi/persian-stopwords.git