سوگل طهباز حسین زاده _ 400522013

عرفان کلندی _ 400522139

```
def load_data(data_path):
    data = []

    with open(data_path, 'r') as file:
        csv_reader = csv.reader(file)

        next(csv_reader, None)

        for row in csv_reader:
            words = preprocess(row[2])
            data.append((words,row[4]))

    return data
```

این تابع یک تابع بر می گرداند که عضو اول آن کلمات تمیز شده ی توییت و عضو دوم آن lable آن توییت است.

سپس فایل csv را می خوانیم و به خط اول آن کاری نداریم چون لیبل ها هستند و به خط بعدی می رویم.

پس به خط اول فایل کاری نداریم و از خط دوم دوم عضو دوم آن یعنی تکست را میگیریم و آن را preprocess می کنیم و کلمات آن را برمی گردانیم. سپس به دیتا تاپل کلمات و لیبل را اضافه می کنیم.

```
def preprocess(tweet_string):

    tweet_string = re.sub(r'http+|www+|https+', '', tweet_string)
    tweet_string = re.sub(r'@+', '', tweet_string)
    tweet_string = re.sub(r'(.)\1+', r'\1\1', tweet_string)
    tweet_string = re.sub(r"\s+", " ", tweet_string)
    tweet_string = tweet_string.lower()

    tokens = word_tokenize(tweet_string)

    words = []
    stop_words = set(stopwords.words('english'))
```

```
    special_chars = set('!@#$%^&*()-_=+[{]};:\'",<.>/?\\|`~0123456789')

    for word in tokens :
        if word not in stop_words :
            valid = True
            for char in special_chars:
                if char == word:
                    valid = False
                    break
            if valid :
                words.append(word)

    return words
```

در این قسمت توییت را تمیز می کنیم پس ابتدا با استفاده از re.sub مقادیری را که نوشته شده است با ” در توییت عوض می کنیم.

در خط اول می گوییم اگر مقادیر http یا www یا https بود و بعد از آن هر چیزی می توانست باشد در توییت آن را با ‘ ‘ جایگزین کن.

در خط دوم user name ها را حذف می کنیم به طوری که اگر اول آن ها @ و بعدش هر چی بود آن را با ‘ ‘ جایگزین کند.

در خط سوم گفته شد اگر کلمه ای چند بار تکرار شد آن را به دوتا تبدیل کن.

در خط چهارم گفته شد اگر چند space پست سر هم بود آن ها را به یکی تبدیل کن.

سپس روی توییت lower را صدا می زنیم که همه جای آن مثل هم شوند وحرف بزرگ و کوچک فرقی نداشته باشند.

سپس روی توییت word_tokenize را صدا می زنیم که کلمات را بر اساس فاصله از هم جدا می کند و اگر کلمه ای باشد که چسبیده به آن یک کاراکتر مثل علامت سوال باشد و بعد از آن فاصله باشد آن دو را هم از هم جدا می کند و داخل tokens می ریزیم.

در مرحله بعد باید stopword ها و کاراکتر ها را از آن ها جدا کنیم.

ابتدا کلمات stopword را می گیریم و در stop_words قرار می دهیم سپس یک فور روی tokens ها می زنیم و چک می کنیم اگر کلمه داخل stop_word ها بود که کاری به آن ها نداریم اما اگر نبود در ابتدا فرض می کنیم valid = true است و سپس یک فور روی کاراکتر ها می زنیم و چک می کنیم اگر کلمه ی ما یکی از کاراکتر ها بود valid را false

می کنیم و از فور خارج می شویم و در اخر اگر valid = true بود آن کلمه را به کلمه ها اضافه می کنیم و در اخر لیست کلمات را برمی گردانیم.

در آخر در خط 80 فایل run.py

```
nb_classifier.train(load_data(train_data_path))
```

زمانی که داده ها را تمیز کردیم آن را به train می دهیم.

فایل template.py :

```
    def __init__(self, classes):

        self.classes = classes
        self.class_word_counts = {}
        self.class_counts = {"positive":0,"negative":0,"neutral":0}
        self.count = 0
        self.likelihood = {}
        self.prior = {}
        self.vocab = set()
```

خط اول کلاس هارا مشخص می کند.

خط دوم یک دیکشنری را مشخص می کند که کلید آن یک کلمه است و value آن یک دیکشنری که مشخص می کند هر کلمه در چند توییت مثبت و منفی و خنثی آمده است.

در خط سوم مشخص می کند در یک کلاس چند تا از داده ها مثبت و منفی و خنثی هستند

خط چهارم تعدا کل توییت های train را نشان می دهد.

خط پنچم و ششم را در بعد تر محاسبه می کنیم.

خط هفتم یک ست است که تمام کلمه های یونیکی که از داده های train می خوانیم را در آن می ریزیم.

```
    def train(self, data):
      for features, label in data:
          self.count += 1

          self.class_counts[label] +=1

          for word in features :
```

```
                self.vocab.add(word)
                if word not in self.class_word_counts:
                    self.class_word_counts[word] =
{"positive":0,"negative":0,"neutral":0}

                self.class_word_counts[word][label] += 1

        for word in self.vocab:
            self.likelihood[word] = {}
            for c in self.classes:
                self.likelihood[word][c] = self.calculate_likelihood(word,c)

        self.calculate_prior()
```

در ابتدا به عنوان ورودی data را می گیریم که یک تاپل است و روی آن فور می زنیم و هر توییتی که میخوانیم یکی به self.count اضافه می کنیم و همچنین class_counts لیبل آن توییت را یکی زیاد می کنیم. سپس یک فور می زنیم روی کلماتی که در توییت هستند و آن را به vocab اضافه می کنیم و اگر اولین باری بود که آن کلمه را می خواندیم باید آن را initial کنیم و در ابتدا مقادیر را صفر می گذاریم و سپس لیبل آن کلمه را یکی زیاد می کنیم.

سپس به ازای کلمه های یونیک فور می زنیم و حساب می کنیم آن کلمه در کلاس مثبت و منفی و خنثی چقدر احتمال دارد. پس likelihood یک دیکشنری می شود که کلید آن آن کلمه است و value آن احتمال کلاس هاست. یک فور روی کلاس ها می زنیم به ازای هر کلاس برای آن کلمه calculate_likelihood را صدا می زنیم و احتمال آن کلمه در آن کلاس را پیدا می کنیم و داخل دیکشنری likelihood قرار می دهیم.

```
    def calculate_likelihood(self, word, label):
        if word in self.vocab:
            return math.log((self.class_word_counts[word][label]+1) /
(self.class_counts[label] + len(self.vocab)))
        else:
            return math.log( 1 / (self.class_counts[label] + len(self.vocab)))
```

فرمول: تعدا باری که آن کلمه در آن کلاس استفاده شده است به علاوه 1 به تعداد کل توییت هایی که آن لیبل را داشتند بعلاوه تعداد کل کلمه های یونیک.

چک می کنیم اگر کلمه داخل وکب باشد حساب می کنیم آن کلمه به ازای آن لیبل چند بار تکرار شده است بعلاوه یک تقسیم بر تعداد کل کلمه هایی که آن لیبل را داشتند بعلاوه تعداد کل کلمه های یونیک و از حاصل لگاریتم میگیریم و آن را برمیگردانیم و اگر کلمه داخل

وکب نبود یعنی تعداد آن کلمه به ازای آن لیبل صفر است و بقیه حاصل را حساب می کنیم و لگاریتم میگیریم و آن را بر میگردانیم.

```
    def calculate_prior(self):
        for c in self.classes:
            self.prior[c] =  math.log(self.class_counts[c]/self.count)
```

ممکن است تعداد توییت های مثبت و منفی برابر نباشند در دو کلاسه لگاریتم تعداد توییب های مثبت به توییت های منفی را میگیریم اما در سه کلاسه تعداد توییت های آن کلاس را برتعداد کل توییت ها تقسیم می کنیم و لگاریتم می گیریم.

فایل run.py :

```
start_time = time.time()
nb_classifier.train(load_data(train_data_path))
end_time = time.time()
elapsed_time = end_time - start_time
```

در اینجا تایم میگیریم تا train کنیم و وقتی تموم شد دوباره تایم میگیریم و از هم کم می کنیم و مشخص می کند چقدر train طول کشیده است

```
print("train compeleted")
print(f"train time :{elapsed_time}")
```

و سپس تابع eval را صدا می زنیم تا دقت مدل را بفهمیم.

```
def eval():
    with open('eval_data.csv', 'r') as file:
        csv_reader = csv.reader(file)

        next(csv_reader, None)
        valid = 0
        total = 0
        for row in csv_reader:
            c = nb_classifier.classify(preprocess(row[2]))
            if c == row[4]:
                valid +=1
            total+=1
        print("eval compeleted")
        print(f"Accuracy of the model: {(valid/total) * 100}")
```

ابتدا فایل csv را می خوانیم و مانند قبل خط اول را لیبل ها هستند رد می کنیم و دو متغیر valid و total قرار می دهیم که مقدار اولیه هر دو صفر هستند که valid آن هایی که درست حدس زده شده و total هم کل را نشان می دهد.

پس هر ردیفی که می خوانیم آن را تمیز یا preprocess می کنیم و کلمات آن را بر میگردانیم و تابع classify را روی آن صدا می زنیم و کلاس آن را به ما می دهد.

در قسمت بعد چک می کنیم اگر کلاسی که به ما داده بود درست بود valid را یکی زیاد می کنیم چون درست تشخیص داده است اما در هر صورت total را یکی زیاد می کنیم. و وقتی تمام شد این درصد را می گیریم و چاپ می کنیم.

در مرحله بعد باید تست را انجام دهیم.

```
def test():
    result = open("result.txt","w+")
    with open('test_data_nolabel.csv', 'r') as file:
        csv_reader = csv.reader(file)

        next(csv_reader, None)
        for row in csv_reader:
            c = nb_classifier.classify(preprocess(row[2]))
            result.write(f"{c}\n")
    result.close()
```

در این قسمت یک فایل تست csv به ما داده شده است که باید لیبل آن ها را پیدا کنیم و در فایل result.txt بریزیم. پس ابتدا یک فایل result.txt می سازیم که اگر آن را داشت باز می کند و در غیر این صورت آن را می سازد. سپس فایل csv را می خوانیم و تکست آن را تمیز می کنیم و آن را classify می کنیم و یک کلاس به ما می دهد و آن کلاس را داخل فایل result می نویسیم و در آخر فایل را می بندیم.

```
    def classify(self, features):
        best_class = None
        max = float('-inf')
        classValue = {}
        for c in self.classes:
            classValue[c] = 0
            for f in features:
                if f in self.vocab:
```

```
                classValue[c] += self.likelihood[f][c]
            else:
                classValue[c] += self.calculate_likelihood(f,c)
        classValue[c] += self.prior[c]

    for val in classValue:
        if classValue[val] > max :
            best_class = val
            max = classValue[val]


    return best_class
```

در این تابع کلمات یک توییت به ما داده می شود و مشخص می کنیم برای چه کلاسی است

در ابتدا مقدار best_class را برابر none قرار می دهیم و max را برابر منفی بی نهایت قرار می دهیم زیرا تمام اعدامان منفی شده است پس نمی توانیم آن را صفر بگذاریم. سپس یک دیکشنری درست می کنیم که در آن مشخص می کنیم هر توییت به ازای هر کلاس چه احتمالی دارد و بین آن ها max می گیریم.

روی کلاس ها فور می زنیم در ابتدا value آن کلاس برای آن توییت را صفر می گذاریم.

به ازای هر کلمه ای که می خوانیم اگر داخل وکب بود likelihood آن کلاس را که حساب کرده بودیم به value آن کلاس اضافه می کنیم و اگر داخل وکب نبود calculate_likelihood آن را حساب می کنیم و به value آن کلاس اضافه می کنیم و به ازای تمام کلمات که جمع کردیم prior آن کلاس را هم به value آن اضافه می کنیم.

وقتی برای همه ی کلاس ها این مقدار را حساب کردیم بین آن ها ماکزیمم را پیدا می کنیم و آن کلاس را به عنوان best_class قرار می دهیم.

به این ترتیب از این تابع می توانیم در توابع دیگر استفاده کنیم تا کلاس توییت مورد نظر را پیدا کنیم.

خروجی کد به شکل زیر است:

train compeleted

train time :8.539040565490723

eval compeleted

Accuracy of the model: 63.801728058208276

test compeleted

train به اندازه 8.539040565490723 طول کشیده است.

دقت مدل به اندازه 63.801728058208276 است .

و در آخر تست انجام شده است و در فایل result.txt قرار گرفت.