تمرین ششم پردازش زبان‌های طبیعی

امیررضا صدیقین

۹۹۳۶۱۴۰۲۴

مراحل ۱تا۵

در این بخش فایل‌های csv به وسیله‌ی کتابخانه‌ی pandas خوانده شده و داخل یک دیتافریم ریخته شده. همچنین ستونی به اسم data ساخته شد که از بهم پیوستن title و text هر خبر بهم تشکیل شده است.

read csvs and make dataframes

```
In [2]: true_news_df = pandas.read_csv("News/True.csv")
        fake_news_df = pandas.read_csv("News/Fake.csv")
```

```
In [3]: true_news_df["data"] = true_news_df["title"] + true_news_df["text"]
        fake_news_df["data"] = fake_news_df["title"] + fake_news_df["text"]
```

سپس هر داده پیش پردازش شده علائم نگارشی و stop_words و کلمات تک کلمه‌ای حذف شدند و عملیات lemmatize برروی کلمات اعمال شد و متن‌های پیش‌پردازش شده به دست آمده است. برای‌هر یک از داده‌ی درست و داده‌های غلط به همراه برچسب دیتافریمی تشکیل شده است.

برچسب ۱ برای داده‌های true_news و برچسب 0 برای داده‌های fake_news.

preprocess data

remove single character words and stop_words and lemmatize each word

```
In [4]: stop_words = nltk.corpus.stopwords.words('english')
        lemmatize = nltk.stem.WordNetLemmatizer().lemmatize #function lemmatize
        def preprocess(data_list):
            data_tokens_list = []
            tokenizer = nltk.tokenize.RegexpTokenizer('\w+')
            for data in data_list:
                data_tokens = " ".join([lemmatize(token) for token in tokenizer.tokenize(data) if len(token)>1 and token.lower() not in s
                data_tokens_list.append(data_tokens)


            return data_tokens_list
```

```
In [5]: true_news_df = pandas.DataFrame({
            "content":preprocess(true_news_df["data"]),
            "lable":1 # lable for True
        })
```

```
In [6]: fake_news_df = pandas.DataFrame({
            "content":preprocess(fake_news_df["data"]),
            "lable":0 # lable for fake
        })
```

مراحل ۶ تا ۱۳

برای استفاده از داده‌ها داده‌های درست و غلط را با هم ادغام شده و سپس از روی آن‌ها داده‌های آموزش و تست جدا شده است. (۲۰ درصد داده‌ها برای تست و مابقی برای آموزش است.)

```
In [7]: news_df = pandas.concat([true_news_df , fake_news_df])
```

```
In [8]: news_df
```

Out[8]:

	content	lable
0	budget fight loom Republicans flip fiscal scri...	1
1	military accept transgender recruit Monday Pen...	1
2	Senior Republican senator Let Mr Mueller job W...	1
3	FBI Russia probe helped Australian diplomat ti...	1
4	Trump want Postal Service charge much Amazon s...	1
...	...	...
23476	McPain John McCain Furious Iran Treated US Sai...	0
23477	JUSTICE Yahoo Settles mail Privacy Class actio...	0
23478	Sunnistan US Allied Safe Zone Plan Take Territ...	0
23479	Blow 700 Million Al Jazeera America Finally Ca...	0
23480	10 Navy Sailors Held Iranian Military Signs Ne...	0

44898 rows × 2 columns

split test and trains

```
In [9]: train_sentences ,test_sentences,train_lables,test_lables = train_test_split(news_df["sentence"],news_df["lable"],test_size=0.2,sh
```

تابعی نوشته شد که الگوریتم دسته‌بندی و داده‌های آموزش و آزمایش را گرفته و مدل را با استفاده از داده‌های آموزشی آموزش می‌دهد و برچسب داده‌های آزمایش را پیش‌بینی می‌کند و صحت، دقت، بازیابی، f1_score و ماتریس درهم‌ریختگی را نمایش می‌دهد.

classification reports

```
In [35]: def report_of_classifier(train_data , train_lables , test_data , test_lables , classifier):
             classifier.fit(train_data,train_lables)
             predicts = classifier.predict(test_data)
             conf_matrix = confusion_matrix(test_lables,predicts)
             print(f"accuracy = {accuracy_score(test_lables,predicts)}\n",
                   f"precision = {precision_score(test_lables,predicts)}\n",
                   f"recall = {recall_score(test_lables,predicts)}\n",
                   f"f1_score = {f1_score(test_lables,predicts)}\n",
                   f"confusion_matrix = [{conf_matrix[0,0]}  {conf_matrix[0,1]}\n",
                   f"                    {conf_matrix[1,0]}  {conf_matrix[1,1]}]"
                  )
```

با استفاده از روش bag_of_words جملات (هم آموزش ،هم آزمایش) به بردارها تبدیل شده است. (برای تبدیل جملات به بردار باید همزمان هم داده‌های آموزش هم آزمایش به الگوریتم‌داده شود و سپس باز به دو دسته‌ی آموزش و آزمایش تبدیل شود.)

در روش bag_of_words به ازای هر مدل از دنباله‌کلمه (مثل unigram یا bigram که در ورودی کلاس تعیین می‌شود) یک ستون و برای هر متن بودن و نبودن آن با ۰ و ۱ نمایش داده‌میشود.

with Bag_of_words

```
In [13]: bag_of_words_vectorizer = CountVectorizer(ngram_range=(2,2))
         vectors = bag_of_words_vectorizer.fit_transform( train_contents.to_list()+test_contents.to_list())

In [15]: train_vectors = vectors[:len(train_contents)]
         test_vectors = vectors[len(train_contents):]
```

با استفاده از الگوریتم SVM خطی و روش bag_of_words عملیات آموزش و آزمایش انجام شد که به صورت زیر است.

with linear svm algorithm

```
In [13]: report_of_classifier(train_vectors,train_lables,test_vectors,test_lables,SVC(kernel="linear"))

accuracy = 0.9789532293986637
 precision = 0.9827748938178386
 recall = 0.9729035272132679
 f1_score = 0.9778142974527526
 confusion_matrix = [4626   73
                     116  4165]
```

که نشان می‌دهد دقت خوبی داشته است و همه‌ی scoreهای آن در یک حدود است که بیان می‌کند در تشخیص اخبار درست و غلط به خوبی عمل کرده و همچنین با یک هسته‌ی خطی به خوبی عمل کرده است. این الگوریتم بسیار زمانبر است.

همچنین با استفاده از الگوریتم naïve_base مدل MultinomialNB و روش bag_of_words عملیات آموزش و آزمایش انجام شد که به صورت زیر است.

with Naive Bayes algorithm

```
In [14]: report_of_classifier(train_vectors,train_lables,test_vectors,test_lables,MultinomialNB())

accuracy = 0.9790645879732739
 precision = 0.9684138246738384
 recall = 0.9883204858677879
 f1_score = 0.9782658959537572
 confusion_matrix = [4561  138
                      50  4231]
```

در این روش نیز عملکرد خوبی داشته است که در تشخیص اخبار درست بهتر عمل کرده است ولی در تشخیص اخبار غلط مقدار خیلی ناچیزی ضعیف‌تر از SVM است. (البته به خاطر تصادفی بودن تقسیم بندی داده‌های آزمایش و آموزش

این مقادیر را می‌توان صرف نظر کرد.) این الگوریتم بسیار سریع‌تر از الگوریتم SVM است و کمی نتیجه‌ی بهتری نسبت به SVM نیز داشته است.

با استفاده از روش tf_idf جملات (هم آموزش ،هم آزمایش) به بردارها تبدیل شده است. (برای تبدیل جملات به بردار باید همزمان هم داده‌های آموزش هم آزمایش به الگوریتم‌داده شود و سپس باز به دو دسته‌ی آموزش و آزمایش تبدیل شود.) در این روش برای هر کلمه میزان تکرار در آن متن نگه داشته می‌شود. البته می‌توان وزن هر کلمه نیز مشخص شود که برای هر کلمه متناسب با میزان کلیدی بودن آن وزن تخصیص داده شود.

with_idf

In [15]:
```
tf_idf_vectorizer = TfidfVectorizer()
vectors = tf_idf_vectorizer.fit_transform( train_contents.to_list()+test_contents.to_list())
train_vectors = vectors[:len(train_contents)]
test_vectors = vectors[len(train_contents):]
```

با استفاده از الگوریتم SVM خطی و روش tf_idf عملیات آموزش و آزمایش انجام شد که به صورت زیر است.

In [16]:
```
report_of_classifier(train_vectors,train_lables,test_vectors,test_lables,SVC(kernel="linear"))
```

```
accuracy = 0.9944320712694877
 precision = 0.9923202234116826
 recall = 0.9960289651950479
 f1_score = 0.9941711354628118
 confusion_matrix = [4666   33
                     17  4264]
```

در این روش الگوریتم SVM خطی عمکرد فوق العاده‌ای داشته است. به دلیل آن که بردارها بعد زیادی دارند و دارای مقادیر غیر صفر و یک هستند این الگوریتم توانسته به دسته‌ها را از هم تمایز دهد و ساختار با هسته‌ی خطی داشته باشد. ولی همچنان این الگوریتم کند است.

با استفاده از الگوریتم naïve_base مدل MultinomialNB و روش tf_idf عملیات آموزش و آزمایش انجام شد که به صورت زیر است.

In [17]:
```
report_of_classifier(train_vectors,train_lables,test_vectors,test_lables,MultinomialNB())
```

```
accuracy = 0.9400890868596882
 precision = 0.9465521355285135
 recall = 0.926652651249708
 f1_score = 0.9364966949952784
 confusion_matrix = [4475  224
                     314  3967]
```

در این الگوریتم بخاطر سادگی بسیار سریع است و دقت خیلی خوبی داشته است ولی به دلیل تنوع در مقادیر مولفه‌های بردارها نتوانسته به خوبی SVM عمل کند.

پس به ترتیب عملکرد به صورت زیر است. (همه‌ی دسته‌بندها در یک حدود عملکردند ولی هر کدام در بخشی بهترند که در برایند کلی و صحت به صورت زیر هستند.

- SVM خطی با روش tf_idf
- Niave bayes با روش BOW
- SVM با روش BOW
- Nive bayes با روش tf_idf

روابط بین معیارها:

ماتریس confusion به صورت زیرتعریف می‌شود که در آن P (positive) برای برچسب ۱ و N (negative) برای برچسب ۰ است.

Predicted class / Actual class	P	N
P	TP	FN
N	FP	TN

همچنین recall از فرمول زیر بدست می‌آید:

(میزان درستی پیش‌بینی کلاس Positive)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Accuracy نیز به صورت زیر است.

(میزان درستی پیش‌بینی هر داده)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Precision نیز به صورت زیر است.

(میزان درستی حدس زدن کلاس Positive در بین همه‌ی حدس‌های کلاس Positive)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$