# به نام خدا

**امیرمحمد کمیجانی ۹۹۵۲۲۰۳۲**

**توضیحات تمرین سری اول**

**نکات :**

* در هر سِل اگر از سورسی استفاده شده تمام سورس ها را قرار دادم.
* همچنین بعضی توضیحات اضافه و بعد از تست کردن به دست آمده اند که به آنها هم اشاره کردم و توضیح کوتاهی داده ام.

## ۱)

ابتدا corpus خود را از دیتاست reuter خوانده ایم که تایپ آن یک لیست میباشد و هر عضو این لیست یک استرینگ میباشد.

```
# Load the Reuters dataset
corpus = [reuters.raw(file_id) for file_id in reuters.fileids()]

type(corpus[0])
```

```
str
```

حال برای قسمت **text preprocessing** ابتدا **lowercase کردن را با استفاده از تابع lower انجام دادیم** که برای داده های استرینگ قابل دسترس است.

**سپس با استفاده از تابع translate و با کمک از سورس قرار داده شده punctuation ها را حذف میکنیم.**

**در قسمت دوم یعنی تابع probabilistic n-gram** ، ابتدا دیکشنری مربوط به مدل ngram را در نظر میگیریم. سپس بر روی هر جمله در corpus یک پیمایش انجام میدهیم و عملیات هایی که لازم است را انجام میدهیم ابتدا عملیات **tokenize کردن کلمات را با استفاده از تابع word_tokenize موجود در nltk انجام میدهیم.**
**سپس عملیات padding را انجام میدهیم و در ابتدا و انتهای جملات <s> را قرار میدهیم.**
**در مرحله بعد با توجه به مقدار n ، مقدار n-1 کلمه پشت کلمه و خود آن کلمه را مورد بررسی قرار دهیم.**

```
('ltaha', 'automotive', 'technologies'),
('automotive', 'technologies', 'corp'),
('technologies', 'corp', 'year'),
('corp', 'year', 'net'),
('year', 'net', 'shr'),
```

به طور مثال این چند نمونه از خروجی تولید شده از document ngrams میباشد که با توجه به اینکه مقدار n را ۳ قرار دادیم خود کلمه و ۲ کلمه قبل از آن مورد بررسی قرار میگیرد.
**در مرحله بعد کلمات را به دیکشنری ngram_model اضافه میکنیم یعنی تعداد آنها را اضافه میکنیم.**

```
'peak': 1,
'34mlndlr': 1,
'modest': 1,
'section': 1,
'submission': 1,
'poolingofinterests': 1,
'decline': 2,
'japanese': 1,
```

به طور مثال این قسمتی از کلمات و تعداد آنها از یکی از جملات corpus میباشد.

**در قسمت بعد باید احتمال را محاسبه کنیم.تعداد کل کلمات را محاسبه میکنیم و احتمال رخداد کلمات را با توجه به کلمات اطراف آن میسنجیم.**

```
'rise': 0.05555555555555555,
'have': 0.018518518518518517,
'after': 0.018518518518518517,
'higher': 0.018518518518518517,
'against': 0.037037037037037035,
'interest': 0.018518518518518517,
'compared': 0.018518518518518517,
'yearonyear': 0.018518518518518517,
'much': 0.018518518518518517,
'last': 0.018518518518518517,
```

نمونه خروجی حاصل از این تابع به صورت این عکس میباشد که نشان دهنده احتمالات است.

**در تابع سوم باید با استفاده از مدل ngram و احتمال رخداد کلمه را پیش بینی کنیم.**

سپس شرط را در حلقه وایل قرار میدهیم که اگر طول متن تولید شده کمتر از مقدار min_length باشد کماکان در حلقه بمانیم و کلمات جدیدی اضافه کنیم و متن جدیدی تولید کنیم.

**در حلقه ، n-1 کلمه قبلی را در نظر میگیریم و در متغیر prefixقرار میدهیم.**

**سپس با استفاده از prefix احتمال رخداد کلمه قبلی را به دست می آوریم که اگر prefix مقداری نداشته باشد از حلقه بیرون می آییم.**

**سپس کلمات را فیلتر میکنیم و اگر احتمال رخداد آنها کمتر از مقدار threshold باشد آنها را فیلتر میکنیم.**

سپس کلمه بعدی را در کلمات فیلتر شده و بر اساس احتمال آنها و با استفاده از تابع

**random.choice()** انتخاب میکنیم.در این قسمت با استفاده از **filter** **words.keys** کلمات فیلتر شده و با استفاده از **weights** احتمال آنها را میسنجیم و بر اساس احتمالات کلمه بعدی را انتخاب میکنیم. سپس کلمه بعدی را در متن تولید شده قرار میدهیم و با **تابع join** جمله را بر اساس متن تولیدی مینویسیم.

*در قسمت تست کردن طول جمله را برابر 20 کردم تا کمی جمله طولانی تری به در خروجی بدهد.

## قسمت parameter , tradeoffs :

در رابطه با مقدار n_value ، هر چه این مقدار بیشتر شود طبیعتا متن با کیفیت بهتری تولید میشود زیرا کلمه بعدی کلمات بیشتری را برای در نظر گرفتن و انتخاب میسنجد و کلمه بهتری انتخاب میشود اما به طبع با افزایش مقدار آن بار محاسباتی نیز افزایش میابد.ب**ه طور کلی بین انتخاب بهتر کلمه یا همان انسجام متن(coherence) و بار محاسباتی یک تریدآف وجود دارد**.

**بهترین انتخاب پارامتر** برای مقدار n همان ۳ میباشد البته مقدار ۴ و۵ نیز میتواند انسجام خوبی داشته باشد اما محاسبات آن بیشتر است.در نتیجه من این مقدار را تغییر ندادم.

در رابطه با مقدار threshold که در نظر گرفته شده است مشخصا هر چقدر مقدار آنرا بیشتر بگیریم میزان randomness با در نظر گیری احتمالات نیز افزایش میابد و مقدار کم آن انتخاب های بسیار محدود و کمی را به ما میدهد که ممکن است قبل از اینکه به مطلوب مسئله برسیم از حلقه خارج شویم و نتوانیم متنی با طول خواسته شده تولید کنیم.

برای پارامتر مقدار min_length طبیعتا افزایش آن باعث میشود که زمان بیشتری

نیاز به اجرای الگوریتم داشته باشیم و محاسبات بیشتری داشته باشیم و همچنین کمبود آن باعث میشود که متن ما بسیار کوتاه باشد و متن خوبی استخراج نکنیم. که تریدآف بین این دو موضوع برقرار است. **برای این پارامتر ، مقدار پیشنهادی ۲۰ را قرار دادم. زیرا زمان کمی صرف شد و متن نسبتا کامل و معناداری تولید شد.**

**insight**:

برای این قسمت باید با توجه به تریدآف های گفته شده و البته اولویت ما در حل مسئله پارامتر را انتخاب کنیم.

**** در برخی خروجی ها ، خروجی جمله به صورت لوپ تکرار میشد ؛ با تغییرات جزئی و تست کردن های مجدد این مقدار تغییر کرد و مقدار مشاهده شده در نوتبوک به صورت لوپ نیست.**

## ۲)

دو تابع train , classify را توضیح میدهم.

در تابع train ، تعداد نمونه ها و فیچرهای هر کلاس را در دو دیکشنری کنترل میکنیم.

سپس تعداد فیچرها و لیبل کلاس ها را بدست می آوریم.

سپس با توجه به تعداد به دست آمده مقدار احتمال بر اساس کلاس و فیچر را محاسبه میکنیم.

**در تابع classify ، به دلیل اینکه ضرب این مقادیر ممکن است مقدار بسیار کوچکی برای ما تولید کند و با معضل underflow مواجه شویم از لگاریتم استفاده کردیم که هم این مشکل هندل شود و هم اینکه محاسبه جمع سریعتر و راحت تر میباشد.**

همچنین مقدار لگاریتم را با ثابت بسیار کوچکی جمع کردیم که اگر صفر شد دچار مشکل نشویم.
در انتها از تابع max استفاده کرده ایم که هر کدام که بالاتر بود انتخاب شود.
در خروجی به این مقادیر دقت رسیده ایم که به مقدار مدنظر نزدیک میباشد.

```
Train Accuracy: 0.954375
Test Accuracy: 0.695
```

به طور کلی با بررسی احتمال یک sentiment ممکن است به مقدار مثبت برسیم اما با بررسی جمله متوجه شویم که در دسته منفی میباشد که این میتواند به دلیل استفاده از لگاریتم باشد که مقادیر نسبت به ضرب مقادیر(که باعث underflow) میشد مقادیر بیشتری بگیرند و با جواب درست تفاوت داشته باشند.
**به صورت کلی میتوان این مشکل را به نوعی lack of context دانست یعنی با استفاده از یک کلمه بگوییم مثبت است اما کانتکست کلی منفی باشد.**

**** مقدار دقت برای train , test در عکس آورده شده در بالا و موجود در نوت بوک به دلیل ران گرفتن مجدد متفاوت میباشد.**