درس : کلان داده

نام و نام خانوادگی : فاطمه توکلی

۴۰۰۱۳۱۰۱۶

تمرین شماره ۰۱

بخش اول MapReduce

۱ـ

الف) اگر داشته باشیم ماتریس $A:\ a_{ij}\ in\ row\ i\ and\ column\ j$ و ماتریس $B:\ b_{jk}\ in\ row\ j\ and\ column\ k$

در مرحله اول MapReduce برای ضرب ماتریس:

حاصل ضرب MN یعنی پیوند طبیعی $B(J,K,W)$ و $A(I,J,V)$ که فقط ویژگی مشترک J را دارد، اگر بخواهیم حاصل ضرب را داشته باشیم به صورت $(I,j,k,V*W)$ است

برای تابع map میتوان عنصر مشترک یعنی j را به عنوان کلید و تاپل $(matrix_name, index, value)$ را به عنوان مقدار داشته باشیم مثلا در اینجا داریم:

j,$(A.I.a_{ij})$

j,$(B.k.b_{jk})$

و در مرحله تابع reduce تمامی تاپل‌ها با کلید یکسان را در نظر میگیریم. و کلید-مقدار زیر را با حذف اسم ماتریس و افزودن ترتیب شاخص‌ها تولید میکنیم:

j,$(I,k,a_{ij}b_{jk})$

در مرحله دوم MapReduce :

برای تابع map برای تمام جفت های به دست امده از قسمت قبل مقدار کلید را حذف میکنیم و خروجی به صورت زیر است:

$(i,k)\ ,\ a_{ij}b_{jk}$

و در قسمت تابع reduce برای هر کلید (i,k) جمع انجام میدهیم و خود مقدار کلید را حذف میکنیم.

ب) خروجی mapper اول به صورت جفت مقدار زیر است:

1, $(M1,1,2)$	1: $(M2,1,1)$
1: $(M1,2,1)$	1: $(M2,2,4)$
2: $(M2,1,3)$	2: $(M2,1,2)$
2: $(M1,2,2)$	2: $(M2,2,3)$

ج) تمامی جفت مقدار قبل از وارد شدن به reducer دوم:

$(1,1) : 2$	$(2,1) : 2$
$(1,2) : 8$	$(2,2) : 2$
$(1,1) : 6$	$(1,2) : 9$
$(2,1) : 4$	$(2,2) : 6$

تمام مراحل به صورت زیر است:

Map:

$1: (M1,1,2)$	$1: (M2,1,1)$
$1: (M1,2,1)$	$1: (M2,2,4)$
$2: (M2,1,3)$	$2: (M2,1,2)$
$2: (M1,2,2)$	$2: (M2,2,3)$

Reduce:

$1: (1,1, 2 * 1)$	$2: (1,1, 3 * 2)$
$1: (1,2,2 * 4)$	$2: (1,2,3 * 3)$
$1: (2,1,1 * 2)$	$2: (2,1,2 * 2)$
$1: (2,2,1 * 4)$	$2: (2,2,3 * 2)$

Map:

$(1,1) : 2$	$(2,1) : 2$
$(1,2) : 8$	$(2,2) : 2$
$(1,1) : 6$	$(1,2) : 9$
$(2,1) : 4$	$(2,2) : 6$

Reduce:

8	17
6	10

۲_

الف) در این قسمت روند کار به این صورت است که ابتدا هر خط جداگانه خوانده میشود و ID که اول که مربوط به کانال تبلیغ شده‌است جدا میشود و سپس به ازای هرکدام از ID هایی که در همان خط وجود دارد یک دوتایی (ID : 1) ساخته میشود که در اینجا ID همان کانال تبلیغ شده است ومجموعه دوتایی های تولید شده خروجی تابع map است که به عنوان ورودی به reduce داده میشود تا بر حسب ID یا کلید جمع زده شود و تعداد تبلیغ شدن هرکانال را برگرداند ، در نهایت مرتب سازی انجام شده و ۵ تا از بیشترین کانال‌های تبلیغ شده را در خروجی داریم:

```
Channel 859 has been advertised 1933 times.
Channel 5306 has been advertised 1741 times.
Channel 2664 has been advertised 1528 times.
Channel 5716 has been advertised 1426 times.
Channel 6306 has been advertised 1394 times.
```

ب) روند کار تقریبا مانند‌ه قسمت الف است و در نهایت تعداد تبلیغ کانال هدف را به عنوان خروجی می‌دهیم:

```
The exchange count for channel 1748 is 130.
The exchange count for channel 5633 is 30.
The exchange count for channel 3469 is 119.
```

بخش دوم Association Rule

۱_

بر اساس اطلاعات داده شده، می توان احتمال ظاهر شدن هر کالا در یک سبد را به صورت زیر تخمین زد:

مورد ۱ در همه سبدها با احتمال ۱ ظاهر می شود.

مورد ۲ در نیمی از سبدها با احتمال ۲/۱ ظاهر می شود.

مورد ۳ در یک سوم سبدها با احتمال ۳/۱ ظاهر می شود.

مورد ۴ در یک چهارم سبدها با احتمال ۴/۱ ظاهر می شود.

مورد ۵ در یک پنجم از سبدها با احتمال ۵/۱ ظاهر می شود.

مورد ۶ در یک ششم از سبدها با احتمال ۶/۱ ظاهر می شود.

مورد ۷ در یک هفتم از سبدها با احتمال ۷/۱ ظاهر می شود.

مورد ۸ در یک هشتم از سبدها با احتمال ۸/۱ ظاهر می شود.

مورد ۹ در یک نهم از سبدها با احتمال ۹/۱ ظاهر می شود.

مورد ۱۰ در یک دهم سبدها با احتمال ۱۰/۱ ظاهر می شود.

برای یافتن پرتکرارترین اقلام با آستانه حمایت ۱٪، باید حداقل تعداد سبدهایی را که یک کالا باید در آنها ظاهر شود محاسبه کنیم. از آنجایی که فرض می کنیم تعداد سبدها به اندازه کافی بزرگ است، می توانیم از فرمول استفاده کنیم:

حداقل تعداد سبد = آستانه حمایت * تعداد کل سبدها

با فرض مجموع ۱۰ میلیون سبد، حداقل تعداد سبدهایی که یک کالا باید در آنها ظاهر شود ۱۰۰۰۰۰ سبد است.

با استفاده از این حداقل آستانه، می‌توانیم پرتکرارترین اقلام را با محاسبه فراوانی تخمینی آنها به صورت زیر بیابیم:

مورد ۱ در همه سبدها ظاهر می شود، بنابراین فرکانس تخمینی آن ۱۰۰٪ است.

مورد ۲ در نیمی از سبدها ظاهر می شود، بنابراین فرکانس تخمینی آن ۵۰٪ است.

مورد ۳ در یک سوم سبدها ظاهر می شود، بنابراین فرکانس تخمینی آن تقریباً ۳۳٪ است.

مورد ۴ در یک چهارم سبدها ظاهر می شود، بنابراین فرکانس تخمینی آن ۲۵٪ است.

مورد ۵ در یک پنجم از سبدها ظاهر می شود، بنابراین فرکانس تخمینی آن ۲۰٪ است.

مورد ۶ در یک ششم از سبدها ظاهر می شود، بنابراین فرکانس تخمینی آن تقریباً ۱۷٪ است.

مورد ۷ در یک هفتم از سبدها ظاهر می شود، بنابراین فراوانی تخمینی آن تقریباً ۱۴٪ است.

مورد ۸ در یک هشتم از سبدها ظاهر می شود، بنابراین فرکانس تخمینی آن ۱۲.۵٪ است.

مورد ۹ در یک نهم از سبدها ظاهر می شود، بنابراین فرکانس تخمینی آن تقریباً ۱۱٪ است.

مورد ۱۰ در یک دهم سبدها ظاهر می شود، بنابراین فرکانس تخمینی آن ۱۰٪ است.

با وجود اینکه آیتم های ۲ تا ۱۰ دارای احتمال کاهشی برای ظاهر شدن در یک سبد هستند، اما فرکانس های تخمینی آنها به دلیل زیاد بودن همچنان قابل توجه است. و اگر بخواهیم مثلا از الگوریتم A-priori استفاده کنیم و از الکو هایی ۱ تایی تا ۹ تایی پیش برویم به ترتیب تمام ایتم ها در بر گرفته میشوند که به نظر درست نیست.

دلیل اینکه توزیع احتمال سبدها برای یافتن قواعد ارتباط مفید نیست این است که اطلاعاتی در مورد وقوع همزمان اقلام در سبدها ارائه نمی دهد. توزیع احتمال فقط به ما می گوید که چقدر احتمال دارد یک آیتم در یک سبد ظاهر شود، اما چیزی در مورد رابطه بین اقلام مختلف به ما نمی‌گوید.

۲_

فرآینده گسسته سازی داده‌ها و پیش پردازش‌های لازم:

ابتدا ستون Time را به علت اینکه حاوی اطلاعات متمایزکننده ایی نیست حذف میکنیم

سپس تمامی ستون های v1,..,v28,Amount را بین [-1,1] نرمالایز کرده‌ایم

با استفاده از pd.qcut برای تمام ستون های نرمالایز شده تقسیم بندی انجام میدهیم و این تقسیم به این صورت است که به جای اینکه توزیع را تقسیم کند در هر قسمت تعدا مساوی از داده‌ها را میریزد در اینجا ما هرستون را به ۱۵ مقدار تقسیم کرده ایم و مقادیر ستون amount را نیز به ۱۵ قسمت تقسیم کرده‌ایم و سپس با استفاده از pd.get_dummies مقادیر هرستون بر اساس دسته آن‌ها به مقدار ۰و۱ نگاشت شدند و همچنین کلاس با دو کلاس normal,fraud جابجا شد و مقدار ۰و۱ به ازای هر دسته گرفته اند. درنهایت همه تغییرات ترکیب شده‌اند و دیتاست جدید به صورت زیر ساخته شده‌است :

The get_dummies function is used to convert categorical variables into dummy or indicator variables. A dummy or indicator variable can have a value of 0 or 1.

	Amount_0	Amount_1	Amount_2	Amount_3	Amount_4	Amount_5	Amount_6	Amount_7	Amount_8	Amount_9	...	V28_7.0	V28_8.0	V28_9.0	V28_10.(
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	(
1	0	0	1	0	0	0	0	0	0	0	...	0	0	0	(
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	(
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	·
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
65448	1	0	0	0	0	0	0	0	0	0	...	0	0	0	(
65449	0	0	0	0	0	0	0	0	0	1	...	1	0	0	(
65450	0	1	0	0	0	0	0	0	0	0	...	0	0	0	(
65451	0	0	0	0	0	0	0	0	0	0	...	0	0	0	(
65452	0	0	0	0	0	0	0	0	0	0	...	0	0	0	(

65453 rows × 437 columns

برای تولید قوانی انجمنی لازم است ابتدا قوانین دوتایی یعنی یک کلاس و یک ستون انتخاب شود که به علت نوع گسسته سازی تمام ساپورت ها مقدار یکسانی دارند.

برای تولید قوانین سه تایی از دو حلقه for استفاده کردیم که به ترتیب هر ستون و تمام ستون های بعد آن را ترکیب میکند و در صورتی که مقدار هر دو آن ها ۱ بود میشمارد در انتها مجموعه از ایتم‌های دوتایی را داریم که برای هرکدام ساپورت را حساب کرده ایم:

```
#calculate 2-frequent item/normal
support_2 ={}
not_include = ['Class_fraud','Class_normal']
for i in range(len(creditcard.columns)):
    for j in range(i+1, len(creditcard.columns)):
        col_i = creditcard.columns[i]
        col_j = creditcard.columns[j]
        if (col_i not in not_include) and (col_j not in not_include):
          freq_rule = sum((creditcard[col_i] == 1) & (creditcard[col_j] == 1))
          if freq_rule != 0 :
            support_2[(col_i, col_j)] = freq_rule

support_2 = pd.DataFrame(support_2.items(), columns=['items', 'support'])
```

نتایج مرتب شده براساس ساپورت به صورت زیر است:

	items	support
39262	(V6_14, V24_14.0)	3734
34042	(V5_14, V6_14)	2730
34312	(V5_14, V24_14.0)	2613
5889	(Amount_14, V2_0)	2562
6173	(Amount_14, V20_14.0)	2388
...	...	...
11140	(V1_11, V28_13.0)	1
11086	(V1_11, V25_0.0)	1
10729	(V1_10, V27_13.0)	1
10327	(V1_9, V27_13.0)	1
11564	(V1_13, V3_13)	1

91298 rows × 2 columns

سپس از بین آیتم‌های مرتب شده ۱۰۰۰ تای اول را به عنوان آیتم های پرتکرار در نظر میگیریم و برای آن‌ها confident و interest را به ازای کلاس نرمال به دست می آوریم:

	items	support	confident	intrest
39262	(V6_14, V24_14.0)	3734	1.000000	0.002597
34042	(V5_14, V6_14)	2730	1.000000	0.002597
34312	(V5_14, V24_14.0)	2613	1.000000	0.002597
5889	(Amount_14, V2_0)	2562	0.999610	0.002207
6173	(Amount_14, V20_14.0)	2388	0.997906	0.000503
...	...	...	...	...
84995	(V20_14.0, V28_12.0)	641	1.000000	0.002597
61206	(V12_2.0, V14_13.0)	641	1.000000	0.002597
49956	(V9_4, V27_13.0)	641	1.000000	0.002597
17932	(V2_14, V12_14.0)	641	1.000000	0.002597
44422	(V8_1, V13_14.0)	640	1.000000	0.002597

1000 rows × 4 columns

سپس به ازای مقادیر ستون confident مرتب میکنیم که نتیجه آن ها به صورت زیر است:

این قسمت جواب سئوال ۵ تا قانون برتر سه تایی به ازای کلاس نرمال را نیز در بر دارد:

	items	support	confident	intrest
39262	(V6_14, V24_14.0)	3734	1.0	0.002597
12317	(V1_14, V28_6.0)	734	1.0	0.002597
8751	(V1_6, V3_12)	733	1.0	0.002597
76410	(V16_14.0, V26_7.0)	733	1.0	0.002597
13119	(V2_2, V4_1)	732	1.0	0.002597

برای کلاس جعلی ایتم‌هایی که ساپورت بین ۴۰۰ تا ۱۰۰۰ داشته‌اند را در نظر گرفتیم که بتوان نسبت درستی برای آن ها از بین کل داده ها به دست اورد:

	items	support
34193	(V5_14, V17_0.0)	999
18397	(V3_0, V18_14.0)	997
18263	(V3_0, V10_0)	996
12151	(V1_14, V16_14.0)	996
11935	(V1_14, V2_3)	995
...	...	...
70345	(V14_12.0, V25_2.0)	400
14799	(V2_6, V12_1.0)	400
79938	(V18_6.0, V26_10.0)	400
10364	(V1_10, V3_7)	400
23892	(V4_0, V11_4.0)	400

8658 rows × 2 columns

سپس آن ها بر اساس دو ستون confident مرتب کرده و ۱۰۰۰ تای اول را برای مرحله بعد در نظر گرفتیم، همچنین ۵ قانون برتر سه تایی برای کلاس جعلی نیز در ادامه امده است:

	items	support	confident	interest
34193	(V5_14, V17_0.0)	999	0.001001	-0.001581
18397	(V3_0, V18_14.0)	997	0.009027	0.006445
18263	(V3_0, V10_0)	996	0.146586	0.144004
12151	(V1_14, V16_14.0)	996	0.000000	-0.002582
11935	(V1_14, V2_3)	995	0.000000	-0.002582
...	...	...	...	...
70345	(V14_12.0, V25_2.0)	400	0.000000	-0.002582
14799	(V2_6, V12_1.0)	400	0.000000	-0.002582
79938	(V18_6.0, V26_10.0)	400	0.000000	-0.002582
10364	(V1_10, V3_7)	400	0.000000	-0.002582
23892	(V4_0, V11_4.0)	400	0.000000	-0.002582

8658 rows × 4 columns

۵ تای برتر:

	items	support	confident	interest
28973	(V4_14, V14_0.0)	402	0.343284	0.340702
28942	(V4_14, V11_14.0)	419	0.319809	0.317227
60503	(V11_14.0, V16_0.0)	449	0.300668	0.298086
52838	(V10_0, V12_0.0)	488	0.299180	0.296598
39473	(V7_0, V18_0.0)	400	0.270000	0.267418

آیتم های سه تایی از ترکیب آیتم‌های دوتایی با یک عنصر مشترک به وجود می آیند و نتیجه براساس ساپورت مرتب شده و به صورت زیر است:

	items	support
0	(V5_14, V6_14, V24_14.0)	2526
56	(V2_0, V20_14.0, Amount_14)	1839
2	(V8_13, V6_14, V24_14.0)	1584
1	(V3_0, V6_14, V24_14.0)	1544
57	(V23_0.0, V2_0, Amount_14)	1528
...	...	...
7028	(V23_0.0, V20_0.0, V1_8)	1
7439	(V1_14, V27_5.0, V7_0)	1
6733	(V2_4, V1_14, V7_0)	1
1392	(V2_0, V23_14.0, V1_8)	1
5614	(V2_0, V20_0.0, V1_8)	1

8833 rows × 2 columns

سپس ۱۰۰۰ تای اول را انتخاب کرده و به ازای کلاس نرمال دو فاکتور confident,intrest را حساب کرده:

	items	support	confident	interest
0	(V5_14, V6_14, V24_14.0)	2526	1.000000	0.002597
56	(V2_0, V20_14.0, Amount_14)	1839	0.999456	0.002054
2	(V8_13, V6_14, V24_14.0)	1584	1.000000	0.002597
1	(V3_0, V6_14, V24_14.0)	1544	1.000000	0.002597
57	(V23_0.0, V2_0, Amount_14)	1528	1.000000	0.002597
...	...	...	...	...
4323	(V9_14, V28_0.0, V1_0)	354	0.994350	-0.003052
3594	(V7_14, V5_0, V6_13)	354	1.000000	0.002597
462	(V8_13, V4_0, V24_14.0)	352	1.000000	0.002597
6474	(V24_14.0, V7_1, V8_14)	352	1.000000	0.002597
1028	(V23_0.0, V7_14, V8_0)	352	0.997159	-0.000244

1000 rows × 4 columns

۵ قانون برتر بر اساس فاکتورconfident ایتم‌های چهارتایی به ازای کلاس نرمال به صورت زیر هستند:

	items	support	confident	interest
2883	(V14_0.0, V2_14, V10_0)	356	0.373596	-0.623807
624	(V3_0, V11_14.0, V2_14)	399	0.333333	-0.664069
4177	(V11_14.0, V14_0.0, V6_0)	375	0.322667	-0.674736
4174	(V14_0.0, V6_0, V10_0)	412	0.293689	-0.703713
2211	(V3_0, V14_0.0, V11_14.0)	506	0.290514	-0.706889

برای کلاس جعلی نیز همین روند تکرار شده و نتایج برحسب ساپورت مرتب شده که به صورت زیر است:

	items	support
4442	(V3_0, V2_14, V6_0)	752
17328	(V17_14.0, V3_0, V18_14.0)	747
4910	(V8_14, V2_14, V5_0)	694
4675	(V17_14.0, V3_0, V6_0)	668
16435	(V17_14.0, V14_0.0, V18_14.0)	634
...	...	...
15071	(V23_0.0, V20_0.0, V1_8)	1
2509	(V1_10, V8_14, V4_14)	1
16193	(V21_14.0, V4_13, V1_9)	1
2707	(V1_10, V2_14, V4_14)	1
18661	(V19_0.0, V7_14, V1_8)	1

21370 rows × 2 columns

در نهایت نیز بر اساس فاکتور گفته شده مرتب شده و ۵ قانون برتر به ازای کلاس جعلی برای ایتم های چهارتایی به صورت زیر است:

	items	support	confident	interest
2883	(V14_0.0, V2_14, V10_0)	356	0.373596	-0.623807
624	(V3_0, V11_14.0, V2_14)	399	0.333333	-0.664069
4177	(V11_14.0, V14_0.0, V6_0)	375	0.322667	-0.674736
4174	(V14_0.0, V6_0, V10_0)	412	0.293689	-0.703713
2211	(V3_0, V14_0.0, V11_14.0)	506	0.290514	-0.706889

برای کلاس جعلی دقت به دست آمده به صورت زیر است:

```
count = ((creditcard['Class_fraud'] == 1) & (creditcard['precision_fraud'] == 1)).sum()
percentage = (count / 492) * 100
percentage
```

```
30.284552845528456
```

که فکر میکنم اگر تعداد دسته های هر ستون را بیشتر کنیم دقت بهتری خواهیم داشت مثلا الان به نسبت زمانی که ۱۰ دسته برای هر ستون داشتم دقت بهتری به دست آوردم اگر بتوانیم هر ستون را به جای ۱۵ به ۲۰ ستون افزایش دهیم احتمالا دقت بهتر خواهد شد اما به علت زمان اجرای آن من نتوانستم اجرا بگیرم(کلب همکاری نکرد)

۱_

یکی دیگر از محدودیت های LSH این است که به تنظیم چندین پارامتر مانند تعداد توابع هش، اندازه جداول هش و آستانه در نظر گرفتن جفتهای کاندید نیاز دارد. یافتن مقادیر بهینه برای این پارامترها می تواند چالش برانگیز باشد و تأثیر قابل توجهی بر عملکرد الگوریتم دارد.

علاوه بر این، LSH می تواند false positive و false negative ایجاد کند، به این معنی که برخی از موارد مشابه ممکن است به عنوان کاندید شناسایی نشوند، در حالی که برخی از موارد غیر مشابه ممکن است به عنوان کاندید شناسایی شوند. احتمال false positive و false negative را می‌توان با افزایش تعداد توابع هش و جداول هش کاهش داد، اما این باعث افزایش پیچیدگی محاسباتی تمام می شود.

علاوه بر این، LSH زمانی موثرتر است که ابعاد داده ها بالا باشد، اما با کاهش ابعاد، می تواند کمتر موثر باشد. این به این دلیل است که با کاهش ابعاد، احتمال نزدیکی دو آیتم به یکدیگر در فضای پیش بینی شده کاهش می یابد و شناسایی جفت های کاندید برای LSH دشوارتر می شود.

در نهایت، LSH یک روش اکتشافی است و هیچ تضمینی در مورد کیفیت نتایج ارائه نمی کند. دقت LSH به کیفیت توابع هش و انتخاب پارامترها بستگی دارد و هیچ محدودیت نظری در میزان خطای الگوریتم وجود ندارد.

مهمترین محدودیت LSH (Hashing حساس به محلی) مبادله بین حفظ شباهت و تعداد موارد false positive و false negative است. LSH به طور بالقوه می تواند برخی از جفت های مشابه را از دست بدهد و همچنین می تواند برخی از جفت های متفاوت را، بسته به پارامترهای انتخاب شده و توزیع داده خاص، تولید کند. کیفیت نتایج LSH به شدت به انتخاب توابع هش و تعداد جداول هش بستگی دارد، که باید به صورت تجربی برای مجموعه داده‌های مختلف و معیارهای شباهت تنظیم شوند. علاوه بر این، LSH قابلیت تفسیر محدودی دارد، زیرا مقادیر هش حاصل مستقیماً با هیچ ویژگی معنی‌داری از داده‌ها مطابقت ندارد.

۲_

Min-Hashing تکنیکی است که برای تخمین شباهت بین دو مجموعه استفاده می شود. با هش کردن هر عنصر در مجموعه به یک تابع هش تصادفی و نگه داشتن حداقل مقدار هش برای هر تابع هش کار می‌کند. مجموعه حاصل از حداقل مقادیر هش، signature مجموعه نامیده می شود.

برای اثبات اینکه Min-Hashing دارای قابلیت حفظ شباهت است، باید نشان دهیم که شباهت Jaccard بین دو مجموعه توسط Min-Hash.signature آنها حفظ می شود. شباهت جاکارد بین دو مجموعه A و B به صورت زیر تعریف می شود:

$$J(A,B) = |A \cap B| / |A \cup B|$$

می‌توانیم شباهت جاکارد بین دو مجموعه را با استفاده از Min-Hash،signature آنها به صورت زیر تخمین بزنیم:

فرض کنید S(A) مجموعه حداقل مقادیر هش برای مجموعه A باشد و S(B) مجموعه حداقل مقادیر هش برای مجموعه B باشد. سپس احتمال اینکه حداقل مقادیر هش A و B برای تابع هش h داده شده یکسان باشد برابر است :

$$P(h(S(A)) = h(S(B))) = J(A,B)$$

احتمال اینکه حداقل مقادیر هش A و B برای یک تابع هش معین h متفاوت باشد 1 - J(A,B) است بنابراین، احتمال اینکه حداقل مقادیر هش A و B برای همه توابع هش متفاوت باشد، این است:

$$(1 - J(A,B))^k$$

که در آن k تعداد توابع هش استفاده شده است. شباهت ژاکارد مورد انتظار بین A و B با استفاده از Min-Hashing به این صورت است:

$$E[J(A,B)] = 1 - (1 - J(A,B))^k$$

با افزایش k، شباهت ژاکارد مورد انتظار بین A و B به شباهت واقعی Jaccard نزدیک می شود. این به این معنی است که Min-Hashing دارای قابلیت حفظ شباهت است، زیرا شباهت تخمینی Jaccard با استفاده از Min-Hashing به شباهت واقعی Jaccard بین دو مجموعه نزدیک می شود.

۳_

هرکدام از ۶ تابع یا تسک علامت گذاری شده در زیر همراه با کامنت گذاری آورده شده است و همچنین نتایج در ادامه امده است

```python
def create_shingle(docs: str, k: int):
    """
    :param docs: Entire documents list
    :param k: Shingle size
    :return: A set of k-shingles
    """
    shingle_set = []
    #TODO-Task1: start your code

    # Iterate over each sentence in the collection of documents
    for sentence in docs:
        # Generate a set of shingles for the current sentence
        shingles = set(sentence[i:i+k] for i in range(len(sentence) - k + 1))
        # Add the set of shingles to a list of shingle sets
        shingle_set.append(shingles)

    #end your code
    return shingle_set
```

```python
#TODO-Task2: start your code

# Create a dictionary to map each shingle in the vocabulary to a unique index
vocab_d = {}
for i, shingle in enumerate(list(vocab)):
    vocab_d[shingle] = i

# Iterate over each set of shingles in the document shingles list
for shingle_set in documents_shingles:
    # Create a sparse vector with a length equal to the size of the vocabulary
    vec = np.zeros(len(vocab))
    # Iterate over each shingle in the current shingle set
    for shingle in shingle_set:
        # Look up the index of the current shingle in the vocabulary dictionary
        idx = vocab_d[shingle]
        # Set the corresponding entry in the sparse vector to 1
        vec[idx] = 1
    # Append the sparse vector to a list of sparse vectors
    sparse_vector_matrix.append(vec)

#end your code
```

```python
#TODO-Task3: start your code

# Generate a random permutation of integers from 1 to `size`
permutation = np.random.permutation(size) + 1

# Create a list of permutation as hash_function
hash_func = list(permutation)

#end your code
```

```python
#TODO-Task4: start your code
for vector in sparse_vector_matrix:

    # Get index locations of every 1 value in vector
    idx = np.nonzero(vector)[0].tolist()

    # Use index locations to pull only positive positions in minhash
    shingles = minhash_functions[:, idx]

    # Find minimum value in each hash vector
    signature = np.min(shingles, axis=1)

    # Append signature to M
    M.append(signature.tolist())

#end your code
```

```python
#TODO-Task5: start your code

# Jaccard similarity
result=len(a.intersection(b)) / len(a.union(b))

#end your code
```

```python
#TODO-Task6: start your code

# Retrieve the signature vector for the target document
target_vector = signature_matrix[target_id]

# Initialize an empty list to store Jaccard similarity scores
similarities = []

# Iterate over the candidate documents
for candidate_id in candidates:
    # Retrieve the signature vector for the candidate document
    candidate_vector = signature_matrix[candidate_id]
    # Compute the Jaccard similarity between the target and candidate vectors
    s = jaccard(set(target_vector), set(candidate_vector))
    # Append the similarity score to the list
    similarities.append(s)

# Sort the similarity scores and get the indices of the top k candidates
top_k_indices = np.argsort(similarities)[-k:]

# Retrieve the indices of the top k candidates
result = [candidates[i] for i in top_k_indices]

#end your code
```

کاندیدها:

```
[1, 4, 5, 13, 19, 25, 34, 41, 45, 48, 49]
```

جمله هدف:

```
Target sentence
The lazy dog is jumped over by a quick brown fox.
```

جمله های کاندید:

```
Candidate sentences
The quick brown fox jumps over the lazy dog.
A lazy dog has a quick brown fox jumping over it.
The quick brown fox leaps over the dog that is lazy.
A quick brown fox is seen jumping over a lazy dog.
The lazy dog is leaped over by a quick and brown fox.
The quick brown fox is leaping over the lazy dog.
The lazy dog is jumped over by a quick and brown-colored fox.
The quick and brown fox is jumped over by the lazy dog.
The quick brown fox jumps over the sleeping dog.
A sleeping dog has a quick brown fox jumping over it.
The quick brown fox leaps over the dog that is sleeping.
```

۵ تا از شبیه ترین ها:

```
Similar sentences:
[19, 41, 45, 13, 1]
```