پاسخ تسک ورودی سامرکمپ تحلیل داده

محمدحسین زارعی | github.com/mhezarei

فهرست مطالب

بخش اول - آشنایی با داده

پیش‌پردازش

برای راحتی کار داده را به دو Dataframe به نام‌های load_df و click_df تقسیم می‌کنیم. البته باید بررسی کنیم که تعداد خانه‌هایی که در هر کدام از این دو دیتافریم مقدار NaN دارند برابر با تعداد سطرهایشان باشند تا رکوردی گم نشود. عکس زیر درستی این امر را نشان می‌دهد:

```
[8]  1  click_df_nan = click_df.isna().sum()
     2  nan_tokens = click_df_nan["tokens"]
     3  nan_offsets = click_df_nan["post_page_offset"]
     4  assert(nan_tokens == nan_offsets == click_df.shape[0])
```

```
[9]  1  load_df_nan = load_df.isna().sum()
     2  nan_indices = load_df_nan["post_index_in_post_list"]
     3  nan_post_tokens = load_df_nan["post_token"]
     4  assert(nan_indices == nan_post_tokens == load_df.shape[0])
```

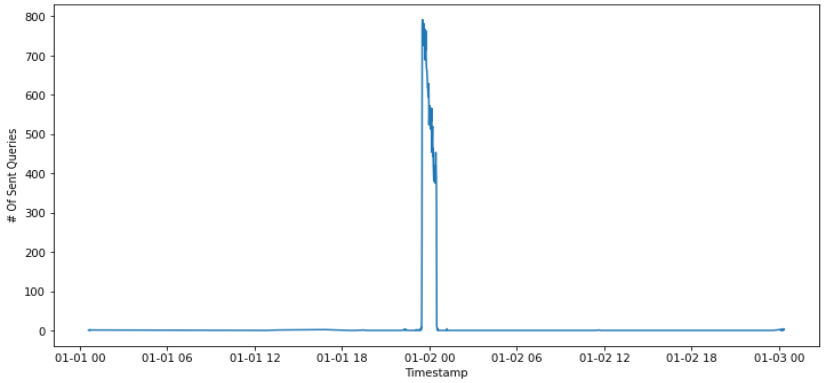
از نظر پیش‌پردازش، در این قسمت دو کار انجام شده است و بقیه‌ی موارد پیش‌پردازش نیز در سوال اول صورت گرفته‌اند. اولین عملیات تبدیل زمان از فرمت داده‌شده به فرمت `"%Y-%m-%d %H:%M:%S"` است. البته از آنجایی که داده در مورد رفتار کاربران در دو روز خاص است، ماه و سال عملاً اطلاعاتی به ما نمی‌دهند. همچنین چون ستون tokens در دیتافریم load_df به صورت string نگهداری شده، آن را به آرایه‌ای از stringها تبدیل کرده و طول هر آرایه را به عنوان ستونی جدید به نام tokens_len نگهداری می‌کنیم.

کاوش‌گری در داده

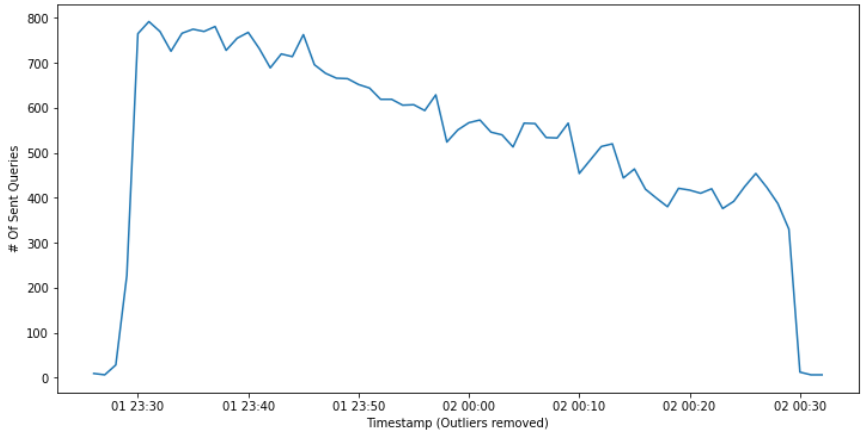
در این قسمت قصد داریم تحلیلی روی داده انجام دهیم با هدف کسب دانش بیشتر در مورد آن. **البته این بخش ارتباطی با پاسخ سوالات ندارد.**

توزیع زمانی کوئری‌ها

مورد اولی که بررسی می‌کنیم زمان ارسال کوئری‌ها به سیستم است. زمان در حالت عادی تا دقت ثانیه نگهداری می‌شود اما برای راحتی کار، دقت به دقیقه کاهش داده شده. نمودار این توزیع را در شکل زیر مشاهده می‌کنید:

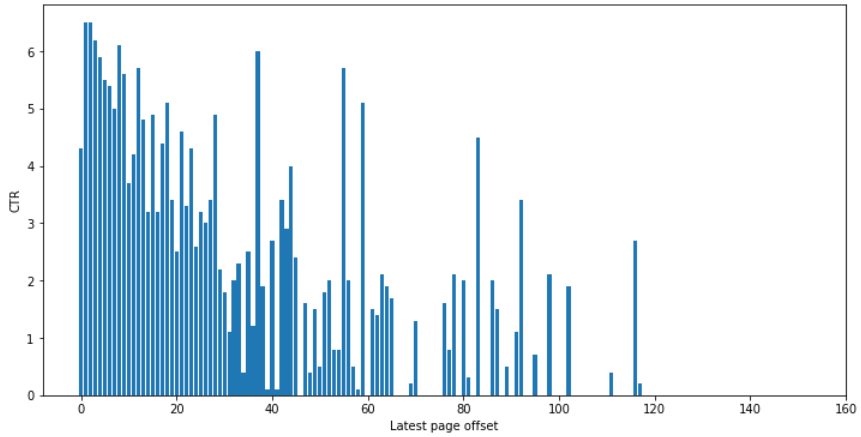


همان‌طور که در شکل مشاهده می‌کنید درصد زیادی (99 درصد!) از کوئری‌ها در یک فاصله‌ی زمانی کوچک وارد شده‌اند. تصویر زیر همین نمودار است اما outlierها در آن حذف شده‌اند:

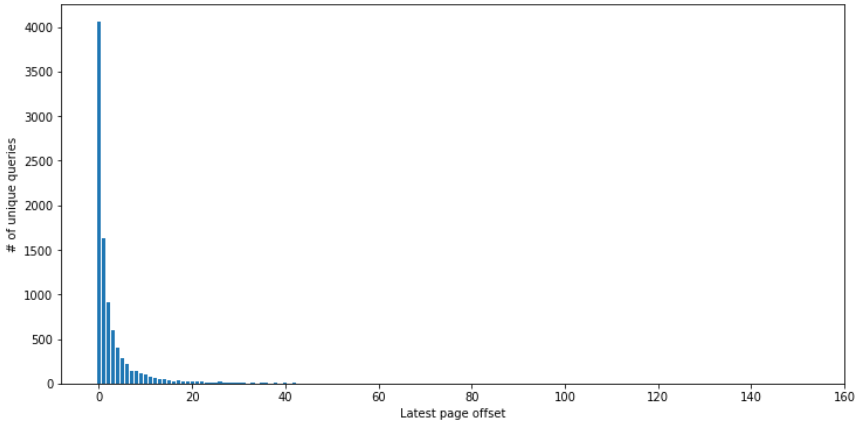


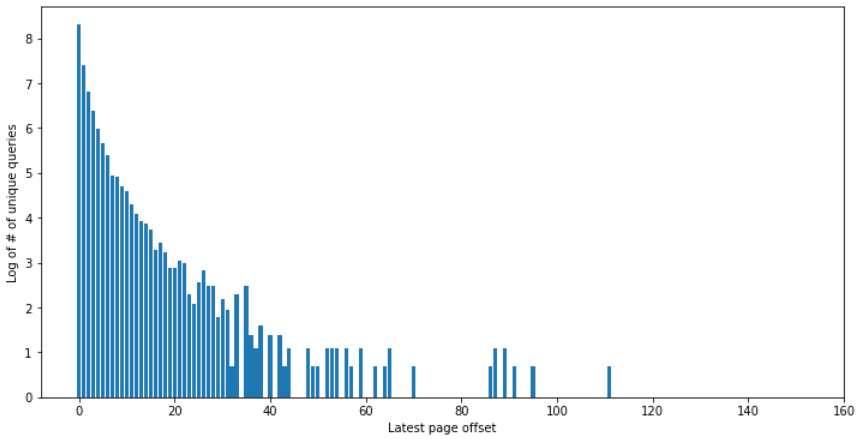
درصد کلیک کاربران با توجه به تعداد پیج‌های لودشده برای آنها

در این قسمت می‌خواهیم بدانیم که با توجه به تعداد پیج‌های لودشده، چند درصد از آگهی‌ها کلیک خورده‌اند. مثلاً در کوئری‌هایی که فقط 5 پیج برایشان لود شده، چند درصد آگهی‌ها کلیک خورده‌اند؟ برای این کار ابتدا دیتافریم load_df را بر اساس تعداد پیج‌های لودشده گروه‌بندی می‌کنیم و سپس بررسی می‌کنیم که برای هر گروه (گروه = کوئری‌هایی که برایشان k پیج لود شده) چند کوئری وجود دارد و بعد از آن از روی دیتافریم click_df تعداد کلیک‌های آن کوئری‌ها را حساب می‌کنیم. در ضمن برای راحتی کار تعداد کل آگهی‌های نشان داده‌شده برای هر کوئری برابر با number_of_pages * 24 در نظر گرفته شده است؛ یعنی اگر مقدار offset یک کوئری 6 باشد، پس 7 پیج برای آن کوئری نمایش داده شده که فرض می‌کنیم در کل 24*7=168 آگهی برای آن کوئری نمایش داده شده. در شکل زیر توزیع درصد کلیک‌ها را مشاهده می‌کنیم:

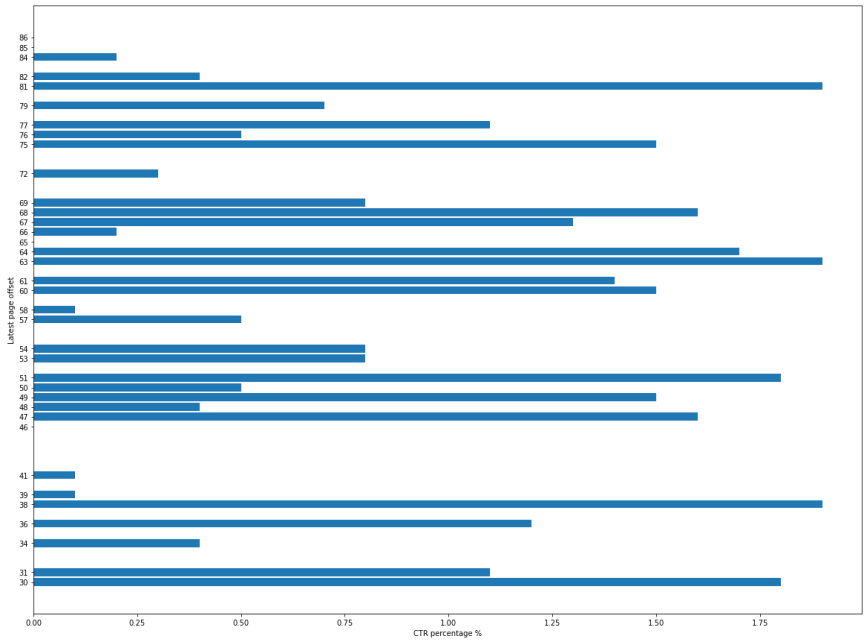


همان‌طور که می‌بینید درصد کلیک با افزایش پیج‌های لودشده کاهش می‌یابد که نشان‌گر این است که یا فرد آگهی مورد نظر خود را یافته ویا از جست‌وجو منصرف شده. در دو شکل زیر (دومی صرفاً لگاریتمی‌شده‌ی اولی‌ست) نیز تعداد کوئری‌ها در هر کدام از این گروه‌ها آمده‌اند:



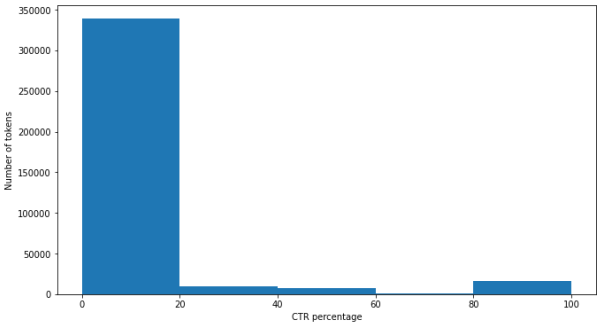


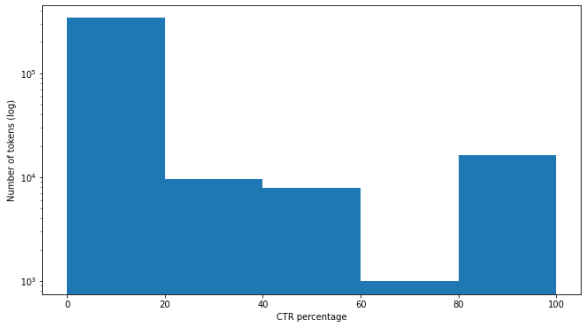
همچنین در شکل زیر نیز گروه‌هایی که نرخ کلیک آنها از 2 درصد کمتر است آورده شده‌اند. می‌توان با استفاده از این گروه‌ها و کوئری‌هایشان علت نبود علاقه و نرخ پایین کلیک را تشخیص داد.



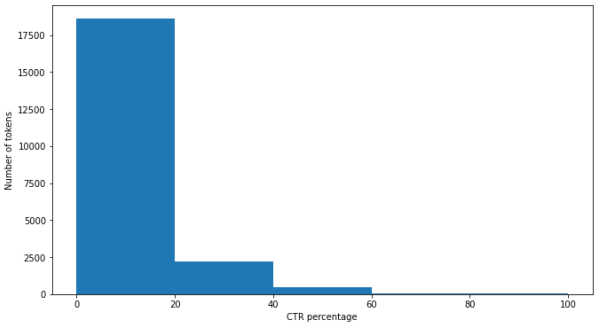
بررسی نرخ کلیک آگهی‌ها (و نه کوئری‌ها)

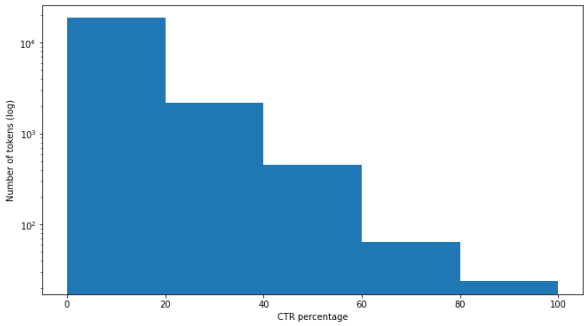
در این قسمت می‌خواهیم بدانیم که نرخ کلیک روی هر آگهی به چه صورت است. برای این کار ابتدا کل آگهی‌های موجود در دیتافریم load_df را استخراج کرده و با توجه به تعداد کلیک‌ها در دیتافریم click_df، به هر کدام از آگهی‌ها درصدی را نسبت می‌دهیم که نشان‌دهنده‌ی نرخ کلیک آن آگهی‌ست. دو نمودار زیر توزیع هیستوگرام نرخ‌ها و لگاریتم آن را نشان می‌دهند:





دو نمودار زیر نیز همین نتایج را برای آگهی‌هایی که حداقل 5 بار نشان داده شده‌اند را نمایش می‌دهد.





با توجه به این نتایج می‌توان آگهی‌هایی که نرخ کلیک پایین و نرخ نمایش بالایی دارند را شناسایی کرده و بهبود داد. همچنین می‌توان از نکات مثبت آگهی‌هایی که نرخ کلیک بالایی دارند بهره برد.

کوئری‌هایی که فقط یک پیج برای آنها لود شده

با دانستن این متریک می‌توان با تقریب خوبی حدس زد که خروجی سیستم پیشنهاددهنده برای آن کوئری خروجی مناسبی بوده چرا که کاربرها نیازی به پیج جدید نداشتند. البته از این موضوع می‌توان این را نیز برداشت کرد که کاربر از نبود نتیجه خسته شده و ادامه‌ی جست‌وجو را نداده. تشخیص این امر از چارچوب داده‌ی فعلی خارج است و می‌توان با داده‌های دیگری مثل داشتن interaction کاربر با آگهی (مثلاً آیا کاربر عکس‌های آگهی را دیده؟ با شماره‌ی داده‌شده تماس گرفته؟ ویا موارد دیگر) به این درک رسید. جدول زیر تعدادی از کوئری‌های مورد نظر را نمایش می‌دهد.

	source_event_id
0	0057b345-8ad7-4c41-9e74-415e39a3140c
1	00796d71-0256-4241-810e-27f946f402ac
2	008faa8d-6648-42de-a76e-e0a7a97be360
3	009e2d57-d038-4268-9ec4-d021e6894667
4	00b3bd08-7d2c-4f1f-ba7b-f9e44012660f
...	...
1673	ff08b22a-c50e-4241-a9b7-d8a485d34b91
1674	ff804753-8238-4b4e-94ad-bbeec59d3aac
1675	ffb96e50-89ea-4a19-b804-b18b08ce1864
1676	ffbc107f-45f5-4dcf-b496-ce421de644a4
1677	ffd325c8-31c7-4e35-a5ed-ee922594e799

1678 rows × 1 columns

نتایج اولیه‌ی کدام کوئری‌ها کلیک نخورده؟

درست است که مرتب‌سازی آگهی‌ها بر اساس زمان صورت می‌گیرد اما به دو دلیل، این متریک کماکان می‌تواند مورد استفاده قرار بگیرد؛ اولی اینکه به طور پیش‌فرض می‌توان فرض کرد که درصد خوبی از کاربران متوجه امر مرتب‌سازی بر اساس آگهی‌ها نمی‌شوند و فرض می‌کنند نتایج اولیه، بهترند. دومی اینکه درصد قابل توجهی از کاربران (و انسان‌ها در کل!) از گشت‌وگذار در نتایج بیزارند و اگر آگهی مورد نظر خود را در نتایج اولیه پیدا نکنند، جست‌وجو را ادامه نمی‌دهند. در جدول زیر کوئری‌هایی را مشاهده می‌کنیم که تعداد بر روی 3 آگهی برترشان کلیک نشده و می‌توان آنها را به عنوان عملکرد بد سیستم پیشنهاددهنده در آینده بررسی کرد.

	source_event_id
0	00142c59-745c-4004-a955-698ddcf1faa6
2	0017b9ef-5903-40f9-a219-85728eb78436
5	0036112e-03f7-4175-ba32-be852dcad535
8	00570968-361a-4b36-a63d-a4a07acbde83
10	005fd78b-bd9b-4075-bda3-1ea0f855c55c
...	...
10639	ffb96e50-89ea-4a19-b804-b18b08ce1864
10644	ffda8068-9c22-46f0-971e-6b2b19c764ea
10645	ffdd48fc-4c05-4c75-bf8c-5cd42729d2b8
10646	ffe3a1f8-c363-4fbb-af5f-0d244b21aea4
10647	ffec4e11-c0a1-4fa4-8613-0979d6f46918

5675 rows × 1 columns

بخش دوم - پاسخ به سوالات

سوال اول

- **وجود مقدار NaN در ستون device_id در دو دیتافریم و ستون post_token در دیتافریم click_df**: به دلیل کم بودن نسبت مقادیر NaN به تعداد کل (تقریباً 3 درصد) و داشتن اهمیت در محاسبات آتی، حذف شده‌اند.

```
[180]  1  load_df.isna().mean() * 100

created_at          0.000000
source_event_id     0.000000
device_id           1.442458
post_page_offset    0.000000
tokens              0.000000
tokens_len          0.000000
dtype: float64
```

```
[181]  1  click_df.isna().mean() * 100

created_at                0.000000
source_event_id           0.000000
device_id                 1.290306
post_index_in_post_list   0.000000
post_token                0.001319
dtype: float64
```

- **وجود سطرهای تکراری در هر دو دیتافریم**: به دلیل تعداد ناچیز حذف شده‌اند اما عملیات حذف می‌بایست قبل از تبدیل ستون tokens به لیست انجام شود.

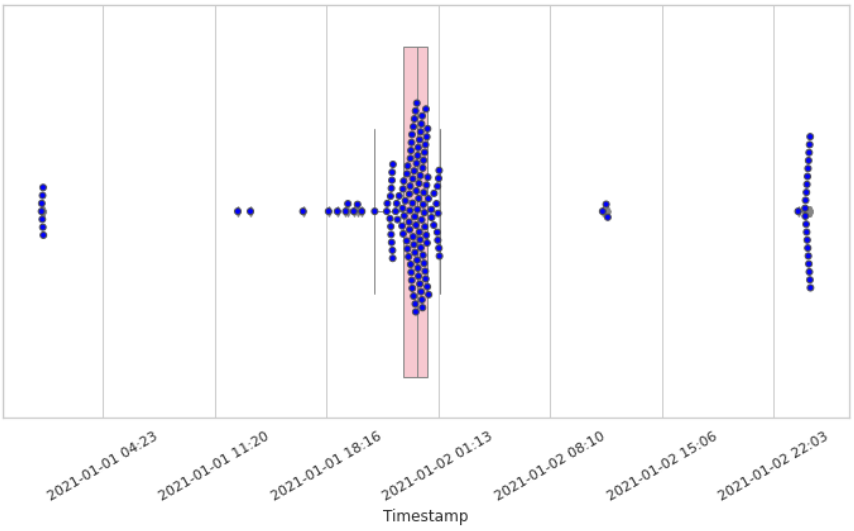
```
[196]  1  print(f"Total number of duplicate rows in load_df = {load_df.duplicated().sum()}")
       2  print(f"Total number of duplicate rows in click_df = {click_df.duplicated().sum()}")

Total number of duplicate rows in load_df = 26
Total number of duplicate rows in click_df = 47
```

- **وجود gap در ستون post_page_offset در دیتافریم load_df برای هر کوئری خاص**: نشان‌دهنده‌ی این است که ممکن است برای برخی از کوئری‌ها، برخی از پیج‌های لودشده در دیتاست آورده نشده‌اند. این کوئری‌ها درصد خوبی از کل کوئری‌ها را تشکیل می‌دهند و به دلیل اهمیت، حذف نشده‌اند. در شکل زیر یک مثال از این موارد را مشاهده می‌کند. در این شکل می‌بایست offsetها از صفر شروع شوند که این اتفاق نیفتاده و gap به وجود آمده. همچنین در کل 6 درصد از کوئری‌ها به این مشکل دچار هستند که 14 درصد از سطرهای جدول را تشکیل می‌دهند.

```
[110]  1  load_df[load_df.source_event_id == gapped_queries[0]].sort_values("post_page_offset")
```

	created_at	source_event_id	device_id	post_page_offset	tokens	tokens_len
4129	2021-01-01 23:30:07	013a873e-0039-45fe-93e8-d6880077b6ce	39OT25_ZRzS9ollvPAVOsg	23	[wXvr2A8C, wXvvGZdT, wXvvG0M0, wXvvWhO-, wXvr2...	24
69059	2021-01-01 23:31:51	013a873e-0039-45fe-93e8-d6880077b6ce	39OT25_ZRzS9ollvPAVOsg	24	[wXv_T-3o, wXvXG0zy, wXvD0JbH, wXvDEPla, wXvjm...	24
69966	2021-01-01 23:33:00	013a873e-0039-45fe-93e8-d6880077b6ce	39OT25_ZRzS9ollvPAVOsg	25	[wXvTmLKA, wXvX2x9N, wXvXGhTf, wXvTmDCP, wXvPm...	24
2722	2021-01-01 23:33:38	013a873e-0039-45fe-93e8-d6880077b6ce	39OT25_ZRzS9ollvPAVOsg	26	[wXvDW5Oi, wXvL2e-U, wXvTksZM, wXvL2Scr, wXvP2...	24
62536	2021-01-01 23:34:31	013a873e-0039-45fe-93e8-d6880077b6ce	39OT25_ZRzS9ollvPAVOsg	27	[wXvHmvll, wXv7V6te, wXvT08vC, wXvXEV20, wXvXU...	24
54121	2021-01-01 23:35:48	013a873e-0039-45fe-93e8-d6880077b6ce	39OT25_ZRzS9ollvPAVOsg	28	[wXvrUNj9, wXvzl2FO, wXv7lpq0, wXv7llb0, wXv7F...	24
109018	2021-01-01 23:37:47	013a873e-0039-45fe-93e8-d6880077b6ce	39OT25_ZRzS9ollvPAVOsg	29	[wXv3VHVf, wXvr1-L6, wXvr0YL4, wXvrlqu-, wXv3F...	24
66083	2021-01-01 23:38:20	013a873e-0039-45fe-93e8-d6880077b6ce	39OT25_ZRzS9ollvPAVOsg	30	[wXmXjoGo, wXvnF7IF, wXvnVMVO, wXvf1Vt2, wXvf1...	24
91600	2021-01-01 23:39:04	013a873e-0039-45fe-93e8-d6880077b6ce	39OT25_ZRzS9ollvPAVOsg	31	[wXvHFPMJ, wXvn1zCP, wXvn1QkX, wXvflH6m, wXvPF...	24

- **نبود برخی از کوئری‌های دیتافریم click_df در دیتافریم load_df**: طبیعتاً تمامی کوئری‌های click_df باید در load_df هم باشند چرا که بدون دانستن اینکه چه پیج‌های برای یک کوئری لود شده، نمی‌توان کلیک‌های آن کوئری را نشان داد. نزدیک به 45 درصد از کوئری‌های دیتافریم click_df این مشکل را دارند که تمامی آنها حذف شده‌اند.
- **وجود outlier در ستون زمان load_df**: با توجه به نمودار زیر به راحتی قابل درک است که چند عدد outlier در ستون created_at از این دیتافریم وجود دارد. همچنین چند عدد از outlierها در تکه‌کد بعد از نمودار قابل مشاهده است.




```
[99]  1  stats = boxplot_stats(query_count.created_at.astype(np.int64))
      2  outliers = [y / 1E9 for stat in stats for y in stat['fliers']]
      3  outliers = [convert_to_date_string(o) for o in outliers]
      4  outliers[:5]
```

```
['2021-01-01 00:36',
 '2021-01-01 00:37',
 '2021-01-01 00:39',
 '2021-01-01 00:40',
 '2021-01-01 00:41']
```

سوال دوم

متریک Dark Query Percentage: برای به دست آوردن این متریک ابتدا آگهی‌هایی که offset آنها برابر با 0 و مقدار tokens_lenشان از 10 کمتر است را فیلتر می‌کنیم و سپس تعداد این کوئری‌ها را به تعداد کل کوئری‌ها تقسیم می‌کنیم و عدد 12.4 درصد به دست می‌آید.

متریک Query Bounce Rate: برای محاسبه‌ی این متریک صرفاً کافی‌ست که مجموعه‌ی کوئری‌های click_df را مجموعه‌ی کوئری‌های load_df کم کنیم. مقدار این متریک برابر 37 درصد است.

سوال سوم

با توجه به این نکته که آگهی‌هایی که از سمت سرور ارسال می‌شوند بر اساس زمان مرتب می‌شوند و نه برحسب مرتبطترین، متریک اول بهترین گزینه در نظر گرفته می‌شود. علاوه بر این نکته توضیحاتی برای هر متریک در ادامه آمده است:

- رتبه‌ی اولین کلیک کاربر: بسیاری از مردم به صورت پیش‌فرض روی اولین مواردی که برای آنها نمایش داده می‌شود کلیک می‌کنند (فرض همه‌ی ما بر این است که موارد اولیه «بهتر»ند) اما این متریک می‌تواند بسیار گمراه‌کننده باشد چرا که معلوم نمی‌کند که آیا کاربر کلیک‌های بیشتری روی موارد بعدی انجام داده یا خیر. همچنین ممکن است که مرتبطترین آگهی یک فرد برحسب زمان در رتبه‌های پایین‌تر باشد که مقدار متریک را کمتر (بدتر) می‌کند ویا کاربر به اشتباه روی رتبه‌های بالایی کلیک کند در صورتی که مقدار مطلوبش نباشد.
- میانگین فاصله‌ی بین رتبه‌های کلیک‌ها: این متریک می‌تواند متریک خوبی برای زمانی که آگهی‌ها برحسب مرتبطترین مرتب می‌شوند باشد و هر چقدر مقدار این متریک کمتر باشد نشان‌دهنده‌ی عملکرد بهتر است اما در این مثال نه.

- اینکه روی سه آگهی اول کلیک شده یا خیر: این متریک هم به دلایلی تقریباً مشابه با متریک دوم (رتبه‌ی اولین . . .) نتیجه‌ی خوبی نخواهد داشت چرا که ممکن است آگهی‌های «بدتر» از نظر زمانی در جایگاه بالاتری قرار بگیرند.
- درصد آگهی‌های کلیک‌شده به کل: بهترین متریک از بین این چهار متریک همین مورد است چرا که ترتیب زمانی آگهی‌ها در نظر گرفته نمی‌شود. هر چه مقدار این متریک بیشتر باشد، از نظر ما عملکرد بهتری انجام شده چرا که نتایج نشان داده‌شده برای کاربر جالب‌تر بوده‌اند.

برای بدست آوردن این متریک ابتدا لازم است تعداد دقیق (برخلاف کار مشابه در قسمت کاوش‌گری در داده) کل آگهی‌های لودشده برای کوئری را داشته باشیم تا تقریب بهتری به دست آوردیم. برای این کار ابتدا فرمول زیر را حساب می‌کنیم:

تعداد کل کوئری‌ها در دیتافریم load_df = تعداد توکن‌های max_offset (که همان آخرین پیج لودشده است) + 24 * max_offset

ممکن است که آگهی‌های موجود در load_df برای یک کوئری به طور کامل لود نشده باشند. برای فهم این امر ماکسیمم اندیس آگهی در دیتافریم click_df را اندازه‌گیری می‌کنیم و بین این عدد و عدد قبلی عملیات max را انجام می‌دهیم. عدد حاصل، تعداد (تقریباً) دقیق کل آگهی‌های لودشده برای هر کوئری‌ست. سپس با تقسیم تعداد کلیک‌های آگهی‌های هر کوئری به تعداد کل آگهی‌های آن کوئری، متریک ومرد نظر را حساب می‌کنیم. میانگین این متریک برای همه‌ی کوئری‌ها برابر با 7.1 درصد است.

سوال چهارم

نشان دادن آگهی به کاربر و عمل کلیک کردن یا نکردن وی را به دید آزمایش برنولی تعریف می‌کنیم. در این صورت عمل «کلیک کردن» به صورت متغیری تصادفی تعریف می‌شود که هر مقدار آن به مجموعه‌ی {0, 1} تعلق دارد و اشتراک چهار متریک گفته‌شده نیز در همین متغیر تصادفی‌ست (در اصل می‌توان همه‌ی متریک‌ها را به توزیع برنولی مپ کرد). همچنین نتایج مرتبط با هر کوئری نیز به صورت یک دنباله از این متغیر تصادفی تعریف می‌شوند که به آن D (مخفف Data) می‌گوییم. هدف ما به دست آوردن تخمینی از احتمال کلیک کردن است که به این احتمال θ یا تتا می‌گوییم. از دو دیدگاه می‌توان این احتمال را تخمین زد که در ادامه هر کدام را بررسی می‌کنیم:

۱. دیدگاه Frequentist

در این دیدگاه تنها با استفاده از روش Maximum Likelihood Estimation به دنبال θ ای می‌گردیم که تابع Likelihood را ماکسیمم کند. ماکسیمم مقدار θ با استفاده از این روش برای توزیع برنولی، برابر با کسر زیر است.

$$\theta_{ML} = \frac{Number\ of\ successes\ (clicked\ tokens)}{Total\ number\ of\ shown\ tokens}$$

پس از آن، فقط با داشتن هر دنباله از آگهی‌های نشان‌داده‌شده (به صورت صفر و یک) می‌توان فرمول گفته‌شده را محاسبه کرد و مقدار θ را به دست آورد. نکته‌ی مهم این دیدگاه این است که هیچ فرضی در آن دخیل نیست و تنها داده‌ی فعلی مشاهده می‌شود و از روی آن تصمیم گرفته می‌شود. مشکل بزرگ این دیدگاه این است که در صورت کمبود داده، مقدار θ مقداری نه‌چندان درست خواهد بود. برای مثال اگر فرض کنیم که مقدار متریک چهارم (آیا روی سه . . .) برابر با مقدار «بله» باشد، پس داده‌ی مشاهده‌شده سه آگهی‌ست که یکی از آنها مقدار 1 داشته و بقیه 0 هستند. در این حالت، مقدار θ برابر با 33 درصد تخمین زده می‌شود که با توجه به پاسخ سوال قبلی، عدد معقولی نیست.

۲. دیدگاه Bayesian

در این روش با داشتن تابع Likelihood و همچنین یک فرض قبلی (به آن Prior می‌گویند)، اقدام به پیش‌بینی مقدار θ به شرط D می‌کنیم (به آن Posterior می‌گویند). عملیات تخمین θ ، به نام Maximum A Posteriori Estimation است که از معادله‌ی زیر به دست می‌آید:

$$Posterior \propto Prior \times Likelihood$$

تفاوت این دیدگاه با مورد قبلی این است که در این دیدگاه **علاوه** بر داده‌ی فعلی، یک فرض قبلی نیز در نظر گرفته می‌شود که باعث بهبود پیش‌بینی خواهد شد. نکته‌ی مثبت این روش این است که داده‌های کم مشکلی به وجود نخواهد آورد اما مشکل این روش نیاز به یک فرض یا همان Prior است. خوشبختانه توزیع برنولی یک Conjugate Prior (علت نام این Prior این است که تابع Posterior و Prior از یک خانواده هستند) دارد که توزیع $Beta(\alpha, \beta)$ است. با در نظر گرفتن موارد گفته شده، اعمال داده‌ی مشاهده‌شده و ساده‌سازی معادله، به عبارت زیر می‌رسیم که همان مقدار تخمین‌شده‌ی θ است:

$$\theta_{MAP} = \frac{Number\ of\ clicks + \alpha - 1}{Total\ number\ of\ shown\ tokens + \alpha + \beta - 2}$$

قبل از اقدام به تخمین θ برای هر کوئری می‌بایست پارامترهای توزیع Prior را مشخص کنیم. این روش به صورت دلخواه انجام می‌شود ولی یک راه خوب می‌تواند استفاده از عدد متریک سوال قبل باشد. می‌دانیم که عدد گفته‌شده برابر با 0.07 بود، پس باید مقادیری را برای α و β انتخاب کنیم که احتمال ماکسیمم در آن نمودار برابر با 0.07 باشد. با در نظر گرفتن مقدار 2 برای α و 15 برای β به شکل زیر می‌رسیم که به نظر شکل مطلوبی‌ست.

پس از انجام این کار، مقادیر α و β مشخص می‌شوند و با مشاهده‌ی هر سری داده مربوط به یک کوئری می‌توان مقدار تتا را تخمین زده و بقیه‌ی متریک‌ها را بر اساس این تخمین، محاسبه کرد.

نکته‌ی قابل توجه در بررسی این دو دیدگاه این است که در دیدگاه Bayesian با داشتن داده به اندازه‌ی کافی (میل دادن تعداد آگهی‌های نمایش داده‌شده و کلیک‌شده‌ها به سمت بی‌نهایت)، مقدار تخمین تتا از α و β بی‌نیاز شده و همان کسر در دیدگاه Frequentist حاصل می‌گردد.