User adoption path data driven framework
for identifying users similar to churned users

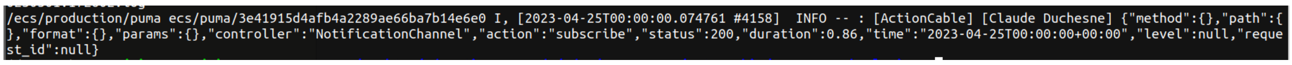
Introduction :

Many users visit digital platform such as Dreeven Platform on a daily basis. Success of such a platform is dependent on user participation. Therefore, understanding the behavior of the user is important. The path taken by any user while using the platform can be referred as the user-adoption path. At times, the path of different users might be similar while at other times this path could be completely different. This makes the task of understanding the behavior of the user complex and difficult. In this work, by comparing the user-adoption path of all the users on the platform with the adoption path of churned users, we want to find out those users that have high chance of dropping out of the platform. If we can do so, we can prevent further churn from taking place by arranging support calls to such users in advance and offering any kind of necessary help that they might need. Doing this will help Dreeven in increasing the length of business that they can do with their subscribed users and thereby prevent any possible decrease in the revenue caused by churning of additional users. Any feedback collected during these calls can be used to further improve the quality of the product offered by Dreeven.

Dataset description :

Real Dataset :

Every time any user logs into the platform and performs any activity on the platform (such as clicks on any feature, uploads a document, views a document), an entry corresponding to such activity gets recorded in the back-end.



```
/ecs/production/puma ecs/puma/3e41915d4afb4a2289ae66ba7b14e6e0 I, [2023-04-25T00:00:00.074761 #4158]  INFO -- : [ActionCable] [Claude Duchesne] {"method":{},"path":{
},"format":{},"params":{},"controller":"NotificationChannel","action":"subscribe","status":200,"duration":0.86,"time":"2023-04-25T00:00:00+00:00","level":null,"reque
st_id":null}
```

Not all entries follow the same format. Therefore, the overall log data is unstructured. In order to understand the data, it needs to be structured.

Dataset pre-processing :

To structure every entry of the log, having an understanding of the attributes specific to this context is necessary. Data from these unstructured entries have been extracted and filled into the following 23 columns to make it structured :

id, datetime, request,params, controller, action, status, duration, time, method, path, format, view, db, level, request_id, session_id, user_id, project_id, company_id, location, platform, client

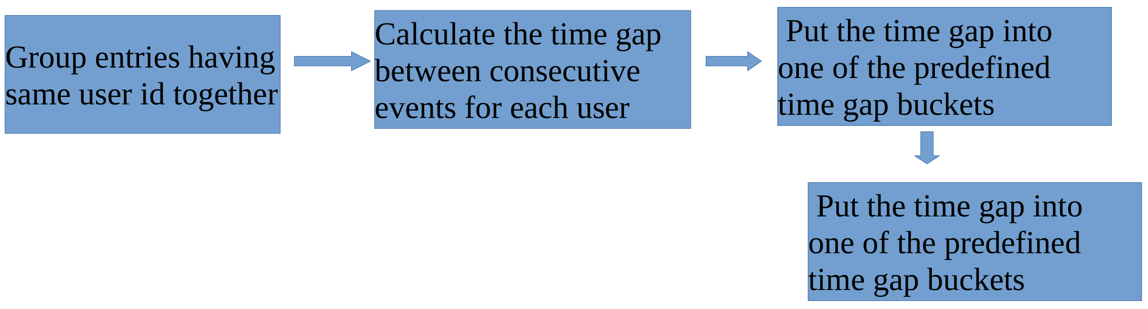
Synthetic dataset and dataset preparation for feature extraction :

Since the real dataset is enormous in size and difficult to process at first, we start the process of extraction of feature for the task of determining the similarity between users in terms of their activity from clickstream dataset (traces of user activity) from the randomly generated sample dataset.

We are interested in every user's type of click events and the time gap between these events.

	user_ids	user_click_events	user_click_event_timestamps
0	0	login	2023-10-05 00:00:00
1	0	settings	2023-10-05 00:10:00
2	1	notifications	2023-10-05 00:20:00
3	2	posting	2023-10-05 00:30:00
4	3	login	2023-10-05 00:40:00
5	3	newsfeed	2023-10-05 00:50:00
6	4	location	2023-10-05 01:00:00
7	2	chat	2023-10-05 01:10:00
8	4	contact	2023-10-05 01:20:00
9	1	project	2023-10-05 01:30:00
10	5	password	2023-10-05 01:40:00

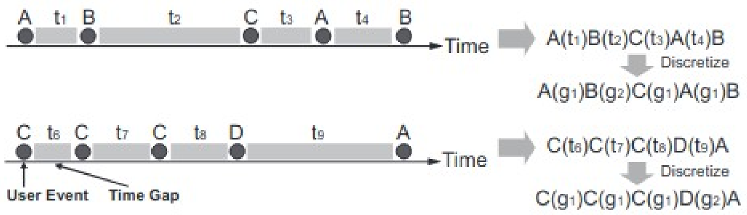
Dataset preparation pipeline :


Group entries having same user id together
Calculate the time gap between consecutive events for each user
Put the time gap into one of the predefined time gap buckets
Put the time gap into one of the predefined time gap buckets

	user_ids	user_click_events	user_click_event_timestamps	time_gap	time_gap (in seconds)	TimeGapBucket	time_gap_events
0	0	login	2023-10-05 00:00:00	None	0	<1 min	g1
1	0	settings	2023-10-05 00:10:00	0 days 00:10:00	600	1 min - 1 hour	g2
2	1	notifications	2023-10-05 00:20:00	None	0	<1 min	g1
9	1	project	2023-10-05 01:30:00	0 days 01:10:00	4200	1 hour - 1 day	g3
3	2	posting	2023-10-05 00:30:00	None	0	<1 min	g1
7	2	chat	2023-10-05 01:10:00	0 days 00:40:00	2400	1 min - 1 hour	g2
4	3	login	2023-10-05 00:40:00	None	0	<1 min	g1
5	3	newsfeed	2023-10-05 00:50:00	0 days 00:10:00	600	1 min - 1 hour	g2
6	4	location	2023-10-05 01:00:00	None	0	<1 min	g1
8	4	contact	2023-10-05 01:20:00	0 days 00:20:00	1200	1 min - 1 hour	g2
10	5	password	2023-10-05 01:40:00	None	0	<1 min	g1

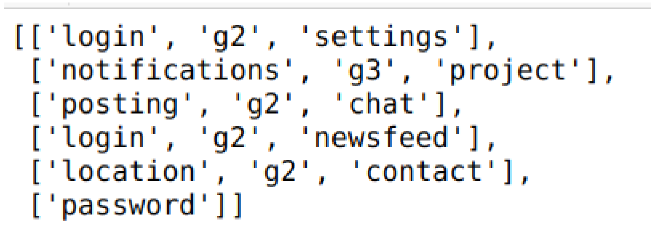
Formatting the clickstream data into a sequence of events :

For every user in the prepared dataset, we express their clickstream activity as a sequence of click event timestamp and time gap event.



Above, we see how the activity of 2 different users are arranged to form a sequence. Here A,B,C,D represent click events where as g1 and g2 represent the time gap event.

Below, we see the activity of all the users (user id : 0-5), expressed in the form of sequence of events.



```
[['login', 'g2', 'settings'],
 ['notifications', 'g3', 'project'],
 ['posting', 'g2', 'chat'],
 ['login', 'g2', 'newsfeed'],
 ['location', 'g2', 'contact'],
 ['password']]
```

For our system, we map the time gap into the following five discrete time buckets: <1s, [1s, 1min], (1min, 1h], (1h, 1day], > 1day, represented by g1, g2, g3, g4 and g5, respectively.

Visually, we can see that u0 and u3 are users which are similar to one another.

Feature extraction and Clickstream similarity graph creation :

Our method is to extract subsequences from the clickstreams as features to compare similarity. More specifically, we formalize a clickstream as a sequence $S = (s_1 s_2 ... s_i ... s_n)$, where s_i is the ith element in the sequence (either a click event or time gap event) and n is the total number of events in the sequence.

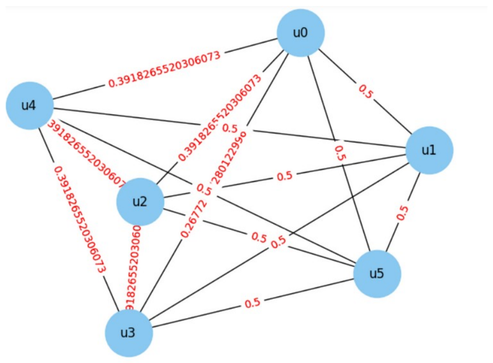
We define T_k as the set of all possible k-grams (k-consecutive elements) in sequence S:
$T_k(S) = \{\text{k-gram} | \text{k-gram} = (s_j\ s_{j+1} ... s_{j+k-1}),\ j \in [1, n + 1 - k]\}$.
To compute the distance between two sequences, we consider both the common k-grams in the two sequences and their count. For S1, S2 and a chosen k, we first compute the set
of all possible k-grams from both as $T = T_k(S_1) \cup T_k(S_2)$. Next, we count the normalized frequency of each k-grams within each sequence l (l=1, l=2) as array $[c_{l1}, c_{l2, ...,} c_{ln}]$ where n = |T|

Finally, their distance is computed as the normalized Polar Distance between the two arrays:
$D(S_1, S_2)$ =

$$\frac{1}{\pi} \cos^{-1} \frac{\sum_{j=1}^{n} c_{1j} \times c_{2j}}{\sqrt{\sum_{j=1}^{n} (c_{1j})^2} \times \sqrt{\sum_{j=1}^{n} (c_{2j})^2}} .$$

The value ranges from 0 to 1, and small distance indicates high similarity between
two clickstreams. We choose Polar distance over other alternatives (e.g., Euclidean distance) because Polar distance is more suitable to handle highly sparse vectors: it compares the “directionality” of the vectors rather than “magnitude”.

For the above sample dataset having six user (user id :0-5), considering 1-gram feature representation of each user’s clickstream, the similarity graph looks like this :



Feature pruning based Clickstream Clustering :

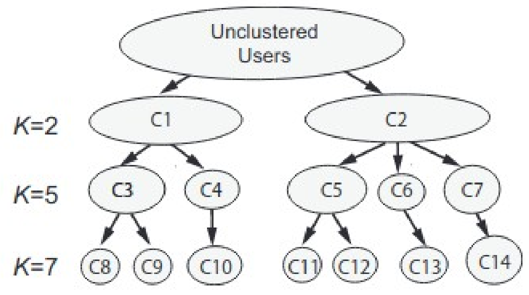


Figure 1. Hierarchy of the behavioral clusters.

A similarity graph dominated by very few features gives little insight on subtle differences between users. The generated clusters may only describe the broadest user categories,while interesting and detailed behavioral patterns remain hidden. We recognize that similarity graph has the capability to capture user behavior at different levels of granularity. We implement iterative feature pruning as a means of identifying fine-grained behavioral clusters within existing clusters, and recursively partitioning the similarity graph.

The key step of feature pruning is finding the primary features responsible for forming the parent cluster. We select features based on Chi-square statistics ($\chi2$) , a classic metric to measure feature’s discriminative power in separating data instances of different classes. For a given cluster, e.g., C1, we measure the $\chi2$ score for each feature based the distribution of users in C1 and those outside C1. We sort and select the top features with the highest $\chi2$ scores. Usually, only a small number of dominating features have high $\chi2$ scores. We automatically select top features by identifying the sweet point (or turning point) in the $\chi2$ distribution.

Understanding the Behavioral Clusters : We can infer the meaning of the clusters based on the selected features during feature pruning phase. A feature is selected because users in this cluster are distinct from users outside the cluster on this particular feature dimension. Thus it can serve as explanations for why a cluster has formed and which user behaviors the cluster encompasses.

In order to achieve our goal, we can use the information about the known churned user and look for the cluster in which it belongs. Other users who fall into the same cluster are the users who have similar activity profile as that of the churned users. These are the users who are likely to drop out of the platform and thus requires reaching out and asking if any kind of help is required by them. This will help reduce the any additional churn and consequently help the business of Dreeven thrive.

Measures taken to ensure efficient computation on the real data :

During the creation of the similarity graph, the real data will have much greater number of features than the synthetic data used. This could result in high computational cost. The approach we have adopted for creating clusters is using feature pruning. This helps us in identifying features that are most dominant (high Chi square value) and thus will help us choose only those features which are useful for our application. This can help us in significantly bringing down the computational cost.

Divisive Hierarchical Clustering using Chi Square Statistics :

Step 1: Calculate Chi-Square Statistics Calculate Chi-Square statistics for each feature based on the distribution of users within the initial single cluster. The Chi-Square statistic measures the association between each feature and the cluster separation. Features with higher Chi-Square values are more relevant for clustering.

Step 2: Rank Features Rank the features based on their Chi-Square values in descending order. This ranking will help you prioritize the most relevant features for cluster separation.

Step 3: Select Features for Divisive Hierarchical Clustering Choose a subset of the top-ranked features to use for divisive hierarchical clustering. The number of features to select depends on your preferences and the level of granularity you want in the clustering. For example, you might choose the top three or four features to start with.

Step 4: Perform Divisive Hierarchical Clustering Now, you can perform divisive hierarchical clustering using the selected features. Divisive hierarchical clustering is a process where you start with all users in a single cluster and iteratively split the cluster into two subclusters based on feature similarity.

1. Begin with all users in one cluster.
2. Use the selected features to determine the most appropriate feature for division.
3. Split the current cluster into two subclusters based on the chosen feature, effectively separating users with different feature patterns.
4. Repeat this process until you have two distinct clusters.

Future work :

1. Pen-paper analysis of feature pruning method for generating clusters
2. Validation of generated clusters similarity through implementation
3. Preparation of real-dataset to work with the implementation
4. Report the results

References :

[1] http://people.cs.uchicago.edu/~ravenben/publications/pdf/clickstream-chi16.pdf