Event-Driven Interest Detection for Task-Oriented Mobile Apps

Fernando Kaway Carvalho Ota(✉), Farouk Damoun, Sofiane Lagraa, Patricia Becerra-Sanchez, Christophe Atten, Jean Hilger, and Radu State

University of Luxembourg, 27 Avenue John F. Kennedy, 2167 Esch-sur-Alzette, Luxembourg
{fernando.carvalho,farouk.damoun,sofiane.lagraa,patricia.becerra, radu.state}@uni.lu, jean.hilger@ext.uni.lu

Abstract. Mobile applications became the main interaction channel in several domains, such as banking. Consequently, understanding user behaviour on those apps has drawn attention in order to extract business-oriented outcomes. By combining Markov Chain and graph theory techniques, we successfully developed a process to model the app, to extract the click high utility events, to score the interest on those events and cluster the groups of interest. We tested our approach on an European bank dataset with over 3.5 millions of user's session. By implementing our approach, analysts can gain knowledge of user behaviour in terms of events that are important to the domain.

Keywords: Interest detection · Behaviour modelling · High utility events

1 Introduction

Nowadays, people depend on mobile applications, simply known as apps, to carry out daily activities including browsing the web, purchasing or checking bank accounts. These activities have attracted attention of different sectors, such as banking. Understanding the behavior of users when browsing mobile apps has increased interest in recent years to identify relevant products and make personalized offers [2]. This growing importance has driven focus of research on the analysis of task-oriented apps. This latter type of apps are defined to be when a user has a prior motivation in mind when starting a session [3], such as using the bank app to execute a wire transfer.

By analyzing navigation steps of the users' click events, we can identify the final user action, what we define as our High Utility Event (HUE). As instance, a user willing to check his bank balance has to navigate through the app until reaching the screen that presents his account. In this case, the HUE is the click event that contains the account balance. Click events consist of a series of ordered events triggered by user interactions when using the mobile app [4].

T. Hara and H. Yamaguchi (Eds.): MobiQuitous 2021, LNICST 419, pp. 582–598, 2022.
https://doi.org/10.1007/978-3-030-94822-1_38

In this sense, many researchers have proposed various techniques to mine HUEs focused on finding the right candidates to the high utility (importance) [5]. However, the main challenge of those investigations is to find the correct sequence of the user's navigation steps to obtain the target action.

In this paper, we propose a novel approach that combines Markov Chain and graph theory techniques to detect users' interest in HUEs a task-oriented mobile app. This approach mainly consists of mapping the whole mobile app by defining the source node based on the first event when the user starts the mobile app. The algorithm verifies all the possible routes, building a tree where the last nodes will be the events with high utility obtaining the target action or events of interest to the user. In the end, clusters are created to highlight the behavior of the users and the sessions by identifying the target action or the product of interest to the user, such as a purchase or a loan.

Problem Statement. Given an events' sequence of a mobile app, how to score the likelihood the user was to execute a certain target action in a session. As instance, a customer of a banking mobile app started to simulate a loan but did not submit a request. In this example, our goal is to identify how close the user was to finish the loan app.

For small apps, a list of target actions, namely HUEs, can be created by domain experts. However, more complex apps can have many candidates for target events, in other words, there can be several relevant actions that has to be filtered in order to provide real behavior knowledge. Additionally, complex apps from bigger enterprises are frequently changing, adding extra layer of manual mapping HUEs and its navigation paths. Usually, the search space is larger as the navigation events tend to appear in higher proportion than the targets.

Besides finding the right HUEs candidates, it is also necessary to detect the chain of events that lead to the execution of the target. By finding this sequence, it is possible to detect the intention of the user when he is navigating in the app in direction towards a certain action. Therefore, we are also investigating the sequential patterns of the app. By merging those research topics, we reach the term High Utility Event Mining.

2 Related Work

In this work, our goal is to find frequent sequential patterns in the session and identify the interesting ones for the business. In this sense, many researchers have focused on developing models for frequent sequential patterns. The research papers [6,7] used the Markov Chains model to identify users' navigation paths on websites. The model focused on analyzing the user's favorite web page without considering a more sophisticated analysis of user behavior. In [8], the researchers propose modeling user trails on the websites. The model obtained promising results in terms of accuracy and mean reciprocal rank, however, the model did not implement event timing modeling to help generate a non-stationary process in the analysis of random shipping lanes. In [9], they analyze the navigational click-stream data, but to social commerce platforms, which diverges from

oriented-tasks app. In [10], they develop a user navigation behavior on a website model. This model focuses on visualizing web data without implementing user navigation path analysis. In [11], they present a framework for modeling the sequential data capturing pathways. In [12], they propose a model for discovering users' navigation. In both investigations, they implement a reduced sample size, affecting the analysis of user behavior and the model's scalability.

An unsupervised clustering method for analysing behavior based timed k-grams approach was proposed in [13]. Their k-grams construction is based on the event name interpolated with a category of a time interval. Then, their clusters are based on the frequencies of the k-grams in the user sessions. The approach was employed to clickstream datasets from Whisper and Renren social network apps, which contain 33 and 55 types of events, respectively. To those amount of event's types, this approach suits well, but it tends to become more computationally expensive as the k-grams construction exponentially increases when the app has a wider spectrum of events. As instance, the bank app we analysed had 311 events in the period we analysed, resulting in over a million 3-gram possibilities to count their frequencies per session, disregarded the time intervals categories. Therefore, this approach becomes unpractical to our use case in terms of processing and human comprehensible cluster visualisation.

One inspiration to our sequence discovery and subsequently graph modelling is found in [14]. In their effort to mine path in web sessions, they created the concept of a via-link to reduce the noise by removing the bigrams and 3-grams that does not reach a minimum occurrence count. To our use case, filtering off the bigrams or 3-grams that does not reach a probability threshold globally would result in removing also real events' transitions as some of them are likely to appear few times in the dataset. For example, some products that are not commonly purchased would disappear in the modelling with the global threshold proposed in this later paper.

The main contributions of this paper can be summarized as follows: First, a new technique is proposed for mapping the whole mobile app by defining the source node based on the user's first action. Second, a tree is built verifying all the possible routes without going through the same path determining the HUE. Finally, a cluster is created to group the behavior of the users in a behavioural perspective.

3 The Dataset

Our analysis was performed on a click event dataset from an European retail bank app. Each button of the app is uniquely named representing click events that are captured through a specific API HTTP call. This call is triggered at every click during user navigation to register the events with a timestamp and linked to an user and session identifiers.

The dataset contains approximately 3.8 million sessions from over 210 thousand different users in a data collection from 31 sequential days from 2021.

Those sessions have around 28 million events among 311 unique ones. Figure 1(a) shows how many sessions a single customer started in the period and (b) presents how many events a single session usually have. For the purpose of keeping the confidentiality agreement with the institution, we replaced the names of the real events by 'e' plus a number ranging from 0 to 311.

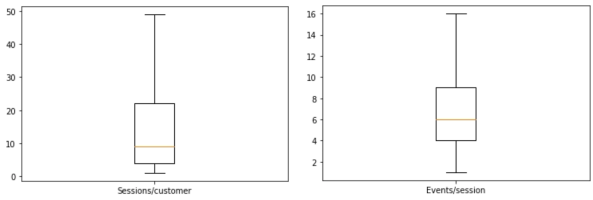

Fig. 1. (a) Boxplot of sessions per customer (b) Boxplot of events per session

3.1 Dataset Issues

An important part of the modelling is data cleansing to avoid analysis on noisy data caused by the occurrence of the following errors. The strategies for addressing those issues are presented in the next sections.

Late Event Registration Due to Disconnections. The user can lose the internet connection and continue using the mobile app. In the bank app we analysed, the strategy is sending the events when the user login again. In this case, the events from the previous session are registered under the current session id, resulting in a incorrect chain of events, as illustrated in Fig. 2.

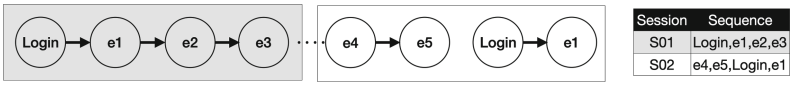

Fig. 2. Late event registration error

Lost Event Registration. There are some cases when the events are lost in the data pipeline. The reason for this varies from problems in the mobile app when sending the requests up to issues on the data flow from the servers to the bank mainframes. In any case, the dataset could contain samples of sequences with missing events as shown in Fig. 3.

Fig. 3. Losing events error

Insufficient Samples for Modelling. In a fully data-driven approach as ours, all the events and its transitions are taken from samples on the dataset. Consequently, if some of the events are not hit or the transitions from one event to another does not have any sample, the model is under represented as in Fig. 4.

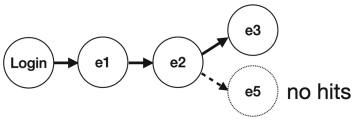

Fig. 4. Lack of event's hits for modelling

4 Methodology

By combining techniques from graph theory, Markov chains, high utility pattern mining, cumulative distribution function scoring and clustering, our approach below is capable of detecting users' interest in a certain event during their navigation in a task-oriented mobile app. Figure 5 depicts the whole processing pipeline of our approach described in the following subsections.

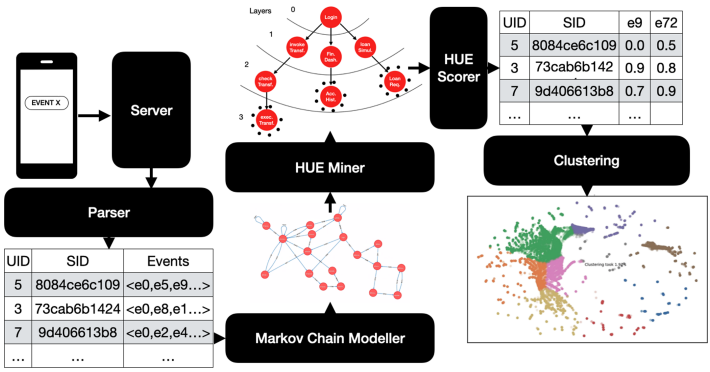

Fig. 5. Overview of the processing pipeline

4.1 Event Definitions

The events considered in this paper are the clicks on the bank app that are captured via API HTTP requests. We first give definitions and notations used throughout this paper, and then give more specific details about our events.

Definition 1. *An event e is a labeled click register generated during the navigation of a user in the app.*

Definition 2. *Let* $E = \{e_1, e_2, \ldots, e_n\}$ *be the set of events.*
$S((uid_i, sid_j), T_s, T_e)$ *denotes the click events sequence of the user* uid_i *in the session* sid_j *between the starting time* T_s *and the ending time* T_e *where* $T_s < T_e$.
$uid_i \in UID$ *where* $UID = \{uid_1, uid_2, \ldots, uid_k\}$ *is the set of users and k is the number of users.*
$sid_j \in SID$ *where* $SID = \{(sid_1, T_{s_1}, T_{e_1}), (sid_2, T_{s_2}, T_{e_2}), \ldots, (sid_l, T_{s_l}, T_{e_l})\}$ *is the set of sessions.*
The click events sequence $S =< (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n) >$, *where* $e_i \in E$, *and* $T_s \geq t_i \leq T_e$ *such that* $t_{i+1} > t_i$.

Table 1 provides an example of the dataset. In practice, the length click events sequences can be very long, and can reach up to a hundred of events per session. In order to mine and characterize causal relations between events, in the next step we introduce the notion of event graph model based on Markov chain as an intuitive graph representation for events.

4.2 Markov Chain Modeller

Our model is based on *n-gram* sequence. An *n-gram* is a contiguous sequence of n events from a given click event sequence. The intuition of the *n-gram* is that instead of computing the probability of an event given its entire history of events, we compute the transition probability of a n set of events occurring in sequence.

Thus, to calculate those transitions, we extract bigrams from the sequences to compute the conditional probability of original event e_o targeting the event e_t. In other words, we approximate it with the probability $P(e_o \rightarrow e_t)$.

Given the events sequence $S_{i,j} =< (e_{1_{i,j}}, t_{1_{i,j}}), (e_{2_{i,j}}, t_{2_{i,j}}), \ldots, (e_{ni,j}, t_{ni,j}) >$ of any user i in any session j, we compute the bigram of the sequence. To compute a particular bigram probability of an event given a previous events, we compute the count of the bigram $C(e_o e_t)$and normalize it with the sum of all the bigrams that share the same first events:

Table 1. Click events dataset.

User ID	Session ID	Click events sequence
U01	S01	<e1, e2, e3, e5, e6>
U02	S02	<e1, e2, e5>
U02	S03	<e1, e3, e5, e7, e3, e5>

$$P(e_o \rightarrow e_t) = \frac{C(e_o e_t)}{\sum C(e_o)} \tag{1}$$

Example 1. Let the following click events sequences of any user and any session from Table 1: $<e_1, e_2, e_3, e_5, e_6>, <e_1, e_2, e_5>, <e_1, e_3, e_5, e_7, e_3, e_5>$. Here are the calculations for some of the bigram probabilities from these sequences.

$P(e_1 \rightarrow e_2) = \frac{2}{3} = 0.66 | P(e_2 \rightarrow e_3) = \frac{1}{2} = 0.5 | P(e_3 \rightarrow e_5) = \frac{3}{3} = 1$

A Markov Chain is constructed from multiple successive events and their occurrence conditional probability. In other way, it is a directed graph that represents successive transitions between events click events sequences through the sessions and users.

Specifically, each vertex in the graph represents an event e_i and each edge (e_i, e_j) indicates that there is a continuous bigram between e_i and e_j. The Markov Chain over a set of type of events V is a weighted directed graph $G = (Vertex, Edge, \gamma)$:

- $Vertex$ is the set of events from E ($Vertex = E$).
- $Edge$ is a set of edges in G. Let e_u and e_v be be two events in $Vertex$. There is an edge $(e_u, e_v) \in Edge$ if and only if there exists a probability with a dependency rule: $e_u \xrightarrow{l_{e_u,e_v}} e_v$.
- γ is the function from Eq. 1 that assigns a probability for each edge (e_u, e_v).

The current transitions between events represent an order of the click events and the behavior of users when they use the app, but also represents the errors presented in Sect. 3.1. In order to reduce the potential noises caused by losing events registration presented in Fig. 3, some of the edges have to be removed. The first strategy for this removal is to exclude when they do not match a minimum transition probability.

Given the weighted directed graph $G = (Vertex, Edge, \gamma)$, and a minimum threshold $min\_support$, the filtered Markov Chain is a weighted directed graph $G_{Reduced} = (Vertex_{Reduced}, Edge_{Reduced}, \beta)$, where:

- $Vertex_{Reduced}$ is the subset of events from E ($Vertex_{Reduced} \subseteq Vertex$).
- $Edge_{Reduced}$ is the subset of edges in $G_{Reduced}$ where $Edge_{Reduced} \subseteq Edge$.
- β is a function that assigns for each edge $(e_u, e_v) \in Edge_{Reduced}$ the probability l'_{e_u,e_v} where $l'_{e_u,e_v} = l_{e_u,e_v} > min\_support$.

In Fig. 6, the minimum support threshold is set to 0.05 as an example. The transition probability from the Login to the event e7 is below this level, therefore, it is removed from the model.

4.3 High Utitily Event (HUE) Miner

Mining the high utility events can be complex in an app that has many events. Mostly because the definition of high utility varies depending on the desired outcomes, as instance, if it is business oriented, then product purchases would

MinSup Threshold > 0.05

Origin	Destination	Counts	Prob.
Login	e2	70	0.7
Login	e3	28	0.28
Login	e7	2	0.02
e2	e6	30	0.6
e2	e7	20	0.4
e3	e7	20	1

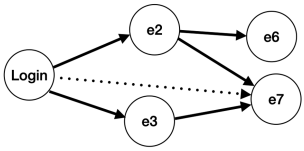

Fig. 6. Markov Chain noise reduction

be relevant, or in a cybersecurity perspective, a password exchange event draw more attention.

Notwithstanding task-oriented apps are likely to have navigation steps until reaching the event to execute the desired action. After executing this action, the user is usually taken back to the main menu or logout the app. If the app is designed in this fashion, consequently, it is possible to mine the HUE in a broad range by assuming the following:

Assumption. In a task-oriented app, HUEs are final states in unique sequences of events.

In this sense, by traversing the graph, the sequences of events can be outlined. However, finding the final states is challenging as it is hard to define the narrowing events of the sequence in an unsupervised way. Intuitively, we could assume that the events before a logout or a return to the main menu could be the candidates for our HUEs. Unfortunately, this assumption fails because the user can logout at any part of the app or return the steps in the sequence until returning to the main menu. Therefore, an algorithm based on this intuition would leave us with many imprecise HUEs candidates.

The solution is to search the graph from a source node, which is the first event when a user start the mobile app, and check on all the possible path are the nodes that can be reached without revisiting a node. Fortunately, the algorithms Depth-first search (DFS) [15] and Breadth-first search (BFS) [16] are techniques that can be employed to this end. Both algorithms traverses the graph and transform it into a tree containing the paths from the root node, the difference is that DFS tries to reach the deepest path while the BFS visit the node neighborhood before moving to the next depth level. To our use case, when a part of the app has multiple options, BFS tends to construct a layered tree where part of those leaves will not have descendants, consequently, outputting wrong candidates for HUEs. On the other hand, DFS group those options in the same level and constructs a tree where the leaves are the final states. By searching the filtered Markov Chain from the login event, DFS outputs a tree where the nodes with out degree equal 0 are our HUEs, as shown in Fig. 7.

DFS would output a perfect list of HUEs if the none of the events is lost during the collection. The cleaning process was done by the minimum support from the bigrams. However, a more frequent "dirty" transition would still persist in the database if their occurrence is above the threshold. Therefore, we adopted

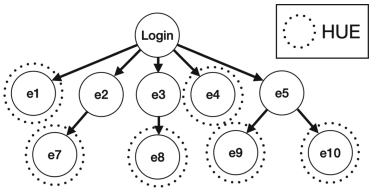

Fig. 7. DFS-based HUE mining

another strategy of creating a path traversal Markov Chain. The path traversal Markov Chain is built by designing a new model of transition analysis, which incrementally creates links (bigrams) and via-links (trigrams), where the edges are weighted according to the relationship between connected nodes. In other words, the co-occurrence of nodes in a session, through a sequence of a mobile app events where a subsequence of size 2 and 3 is considered a link and a via-link of a session, respectively. By setting an additional minimum support for the via-links, the Algorithm (1) will remove the misplaced transitions as double event loss occurrence is less likely to happen then a single loss. In Algorithm (1), we show the graph construction and cleaning as a preprocessing phase of the DFS algorithm in order to discover the data-driven HUEs.

4.4 HUE Scorer

For HUE scorer computation, we consider the Markov Chain as a weighted directed graph (G, w) where $G = (V, E)$ is a directed graph, V is the set of events in the Markov Chain, and E is a set of edges in G where the edge are the connections between events in the Markov Chain. $w : E \to \mathbb{R}_{\geq 0}$ is a non negative weight function. The weight represent the probability between two events in the Markov Chain.

The scoring method is derived from the definition of interest into a HUE. The more close the user is to hit a certain HUE in a session, the more interested he is into executing this action. To score the proximity, we used the Dijkstra algorithm [17] to calculate the distance of all the events to the HUEs in the Markov Chain.

This distance, weighted by the transition probability, represents the score of interest into an HUE. As the Dijkstra algorithm measures the distance to the target node, a lower score means a higher interest into the HUE.

Algorithm 2 represents the computation of the shortest distances of the events to the HUE. It is based on the Dijkstra algorithm on the weighted graph. The inputs are the HUEs and the weighed directed graph constructed from the Markov chain. The algorithm computes the shortest distances from each vertices to the HUE. The output is matrix of weighted distances to the HUEs.

To achieve a score of interest a session in relation to a certain HUE, every session has to be mapped from the event to its distance. The minimum value of this map represents the closest the session was to the HUE.

Algorithm 1. Path Traversal Markov Chain Construction

Input: A collection of app session traces S, η minimum support for links and κ minimum support for via-links

Output: Path Traversal Markov Chain

```
1: unigram_dict ← dict()
2: link_dict ← dict()
3: via_link_dict ← dict()
4: for each s in S do
5:     if length(s) < 3 then
6:         break;
7:     for i = 0; i < s.size() − 2; i + + do
8:         v1 ← s[i]                                          ▷ 1st vertex
9:         v2 ← s[i + 1]                                      ▷ 2nd vertex
10:        v3 ← s[i + 2]                                      ▷ 3rd vertex
11:        if v1! = v2 then
12:            unigram_dict[v1]+ = 1
13:            link_dict[(v1,v2)]+ = 1          ▷ create new key in dict or increase by 1
14:            if v3! = v2 then
15:                via_link_dict[(v1,v2,v3)]+ = 1
16: for each (v1,v2,v3) in via_link_dict.keys() do
17:     if via_link_dict.get((v1,v2,v3) / link_dict.get((v1,v2) > κ then
18:         if link_dict.get((v1,v2) / unigram_dict.get(v1) > η then
19:             G.setEdge(v1, v2)
20:         if link_dict.get((v2,v3) / unigram_dict.get(v2) > η then
21:             G.setEdge(v2, v3)
22: while v = G.vertex() do
23:     if !v.isConnected() then
24:         G.removeVertex(v)
25: return G
```

4.5 Clustering

Clustering the sessions or users based directly on the distances will output improper clusters as bigger the distance is, the lower is his interest to the HUE. Hence, an inversion step is required before running the clustering algorithms. The final score can be calculate in a simple way by applying MinMax function in $score = 1 - MinMax(dist(u))$.

To decide over the clustering algorithms, the size of the app has to be considered. If the app is too big with many HUEs, the amount of cluster tends to rocket as the possibilities are 2 to the HUEs power. Normally, the domain experts are not looking to extract groups of all the mixed HUEs, instead, they should select a group of correlated HUEs.

Algorithm 2. Distance of all events to the HUE.

Input: (G, w) and v. (G, w) an edge-weighted graph (G, w) where $G = (V, E)$ is a graph and $w : E \rightarrow \mathbb{R}_{\geq 0}$ is a non negative weight function. v is the the target vertex.

Output: $dist(u)$ the distance from u to v where v is the HUE.

1: $dist(v) \leftarrow 0$ and $dist(u) \leftarrow +\infty$ / $u \neq v$
2: Queue Q of all vertices in G using $dist$ as the key.
3: **while** $Q \neq \emptyset$ **do**
4: $u \leftarrow min\{dist(x)|x \in Q\}$
5: Remove $u \in Q$
6: **for each** z adjacent to u and in Q **do**
7: **if** $dist(u) + w((u, z)) < dist(z)$ **then**
8: $dist(z) \leftarrow dist(u) + w(u, z)$
9: update $z \in Q$
10: **return** dist

Considering this and the most common clustering algorithms K-Means and DBSCAN, for this type of data, DBSCAN seems to be better as it is not required to specify the numbe of clusters, but K-Means will perform better if the amount of HUEs to be clustered is larger [18].

5 Experiments

5.1 Environment

Our experiments were performed in a cluster with 4 nodes, 128 cores and 1 TB of RAM, running Spark 2.3.0. The parser and the n-grams extraction were implemented in Scala 2.11. The clustering, Algorithms 1 and 2 were coded in Python 3.

5.2 Parsing

The first effort of the work was parsing the API calls to extract the events and their timestamps. The calls persisting in HDFS RDDs consist of "user id, session id, call name, timestamp of the call and message" fields. UID and SID can be retrieved directly by querying the table. However, the message fields from the calls that register the events are the ones we are interested. By filtering those calls, we extract the messages' fields persisted in JSON files that are parsed to fetch the labels and click timestamp from the events, resulting in 2 new fields. Then, the resulting filtered dataset is grouped by the user and session ids. The events with the same user and session id are consolidated in a vector ordered by their timestamps. To fix the date issue 2 from the Sect. 3.1, we concatenate the first part of the events vector, trimmed at the login event, to the previous session in the call timestamp order. The resulting parsed dataset with the columns of Table 1.

5.3 Modelling the App

By iterating over the vectors of events per session to extract the bi-grams and calculating the transitions probabilities with the formula provided in Sect. 4.2, a "dirty" Markov Chain could be plotted. To clean the noisy transitions, we had to set a minimum support threshold based on their probabilities. However, as the threshold increases, we start to lose events and transitions, as seen in Fig. 8. This poses a challenge on how to calibrate the threshold without eliminating the real transitions. We expected that a visual indication would guide the estimation of the threshold.

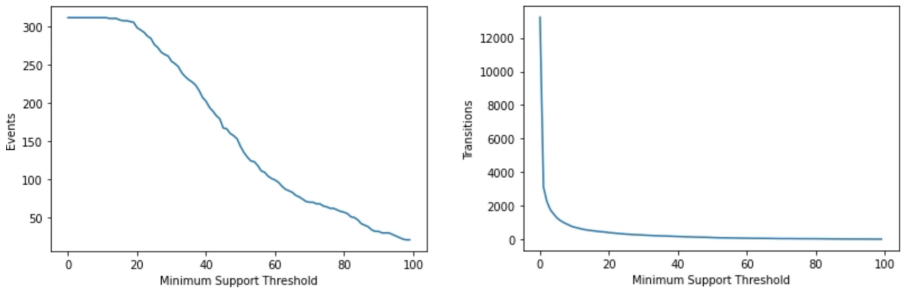

Fig. 8. Threshold distribution

At first glance on the events perspective, it is possible to see that 15% is a maximum threshold in order not to ban events from the model. But taking this 15% means that if the app has a menu with more than 10 possibilities, then at least one of this transitions would be less than 10%.

Another approach would be trying the elbow method for the transitions. Even though, there is an elbow in the right Fig. 8, the precise calibration is not yet straight forward. Actually, to determine this threshold, we had bank experts mapping the path for 7 HUEs, ranging from infrequent to frequent ones. By sliding the threshold in the range of the elbow, we stopped when we start to lose real transitions from the login to the HUE. The losing tendency is bigger to the events that are more infrequent as the transitions probabilities are lower until reaching those events. Finally, we reach a threshold of 2.75%, which seems to be the middle point of the elbow, and then the Markov Chain can be plotted, as the sample Fig. 9.

5.4 Mining the HUEs

The challenge of this task is again finding a minimum support threshold, but this time for the via-links required by Algorithm 1, besides the threshold already found. For this task, we mapped the via-links of the 7 expert HUEs. By identifying the undesired via-links and their probabilities on those HUEs, we reached

the second threshold of 18.6% that is able to remove them. After running the algorithm and applying the output to DFS, 58 HUEs were discovered on top of the 312 events.

For the sake of fitting the output of the algorithm with the DFS in this paper, Fig. 10 illustrates only a subtree containing 42 navigation events leading to 22 HUEs, representing approximately one sixth of the whole tree. Notice that the navigation events are not ordered by their ocurrence when in the same edge, but, it means that those are the multiple clicking options at a certain screen, such as choosing a type of loan.

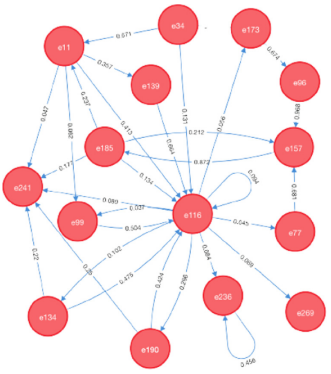

Fig. 9. Frequent Markov Chain

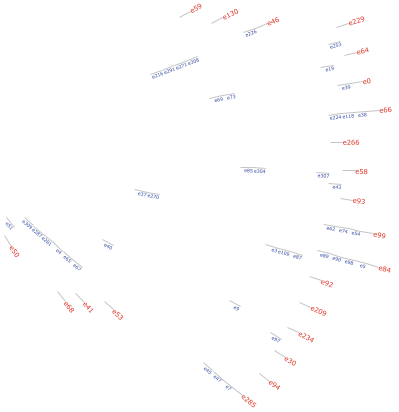

Fig. 10. Mined HUE subtree

5.5 Scoring HUEs

The HUEs are then submitted to the Algorithm 2, outputting a matrix with the weighted distance to all the events to the HUEs. In sequence, every column becomes a dictionary to translate the sessions events into the calculated distances. The minimum distance of each HUE for every session row becomes a new column, generating a frequency matrix for all the UIDs and SIDs.

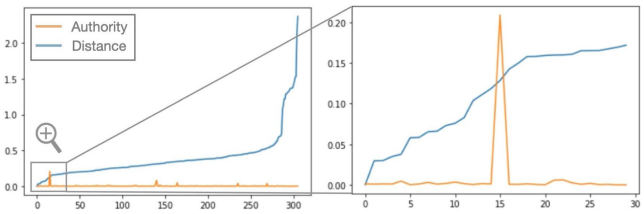

Fig. 11. HUE distances vs HUE authority

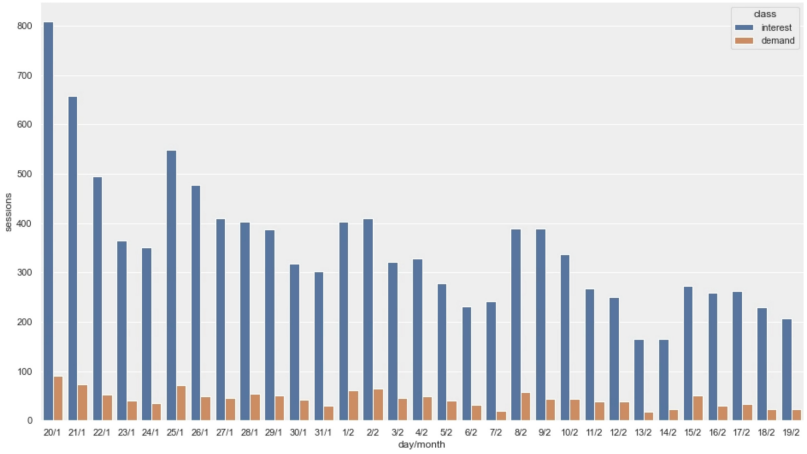

Fig. 12. Specific product interest and demand

This distance can be used a score to highlight the interest of a user in a HUE. To this end, the sessions that have distances below a certain level for a given HUE are the ones that shows interest and when this distance is 0, the user has actually demanded the HUE. In our experiments, we saw that the authority score from the hits algorithm [1] provides indication for determining the upper boundary for the HUE distance. In Fig. 11, it is possible to visualise the maximum distance for personal loan interest that can be set when the authority score raises drastically.

By setting this boundary, the session rows with distance above the level can be filtered off, resulting in a dataset that represents all the sessions interested in a certain HUE. Figure 12 illustrates the distribution of the interest and the actual demand for one of the products in the period we analysed.

5.6 Clustering by HUEs

Our clustering approach aims to identify the behavior of the customers for marketing purposes on the period analysed. To this task, we used the scores of the 7 previously selected HUEs. In order not to disclose any information about the bank's customers, we are not presenting the numbers or the HUEs of the clusters. Still, to ease readers understanding of the clusters, we divided those 7 HUEs in 2 groups: (1) investment (i.e. stock market operation) with 4 products and (2) expenses with 3 products (i.e. loan).

First, we grouped the sessions of the user keeping the maximum value of each HUE score. Second, Principal Component Analysis reduces the dimensionality of the matrix. In sequence, we applied K-Means with 7 HUEs to try to identify the products. The labels of K-Means are used by a Decision Tree Classifier to identify which HUEs are the most significant to each cluster. Finally, we runned DBSCAN to try to extract the total amount of clusters.

Figure 13(a) presents the output of K-Means, where it is possible to see a clear separation between the investment and expenses products. In opposition, the outcome of DBSCAN of Fig. 13(b) does not gives any visual information. Yet, DBSCAN generated 98 clusters on top of $2^7 = 128$ potential HUEs combinations.

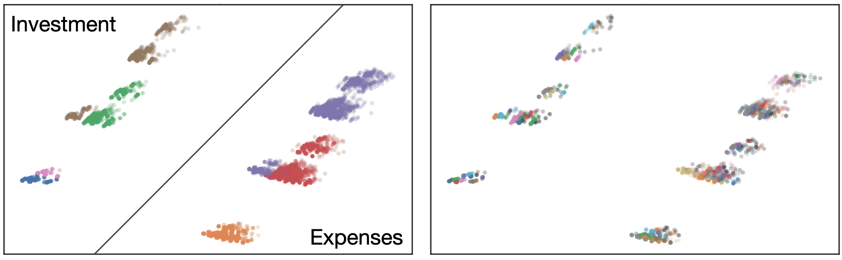

Fig. 13. Clusters found by (a) K-Means and (b) DBSCAN

To our goal, K-Means is more human explainable, while, DBSCAN resulted in better product agglutination, allowing us to see the combination of products among the customers.

6 Limitations

The main bottleneck in our work is the scoring method with time complexity $O(|E| + |V|log|V|)$ [19]. The use of random or grid search does not scale well regarding our approach and the data size, while the influence of η and κ hyper-parameters is undoubtedly important and we have no guarantee to explore the hyper-parameter space without putting human-in-the-loop, experts.

On the other hand, computing clustering accuracy is a limitation as to perform this computation we would need to have the bank experts labelling all the app events.

7 Conclusion and Future Work

The work presented here succeed in automatising the process of detecting interest in a HUE. Our approach can fully map task-oriented mobile apps in a data-driven fashion. The novel HUE mining approach can be extended to other mobile apps domains. Our clustering in combination with scoring techniques are capable of highlighting behavior of users and sessions.

In the future work, we intend to implement the whole process in real-time setup for scoring client sessions and handle the issues discussed in Sect. 3.1 in real-time.

References

1. Kleinberg, J.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–32 (1999). https://doi.org/10.1145/324133.324140
2. Gan, W., Lin, J.C.W., Fournier-Viger, P., Chao, H.C., Tseng, V.S., Yu, P.S.: A survey of utility-oriented pattern mining. IEEE Trans. Knowl. Data Eng. **33**(4), 1306–1327 (2021). https://doi.org/10.1109/TKDE.2019.2942594
3. Raphaeli, O., Goldstein, A., Fink, L.: Analyzing online consumer behavior in mobile and PC devices: a novel web usage mining approach. Electron. Commer. Res. Apps **26**, 1–12 (2017)
4. Bucklin, R.E., Sismeiro, C.: Click here for internet insight: advances in clickstream data analysis in marketing. J. Interact. Mark. **23**(1), 35–48 (2009)
5. Truong-Chi, T., Fournier-Viger, P.: A survey of high utility sequential pattern mining. In: Fournier-Viger, P., Lin, J.C.W., Nkambou, R., Vo, B., Tseng, V.S. (eds.) High-Utility Pattern Mining. SBD, vol. 51, pp. 97–129. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-04921-8_4
6. Schneider, F., Feldmann, A., Krishnamurthy, B., Willinger, W.: Understanding online social network usage from a network perspective. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, pp. 35–48 (2009)
7. Benevenuto, F., Rodrigues, T., Cha, M., Almeida, V.: Characterizing user behavior in online social networks. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, pp. 49–62 (2009)
8. Borisov A, Wardenaar M, Markov I, De Rijke M. A click sequence model for web search. In: 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, pp. 45–54 (2018). https://doi.org/10.1145/3209978.3210004
9. Kumar, A., Salo, J., Li, H.: Stages of user engagement on social commerce platforms: analysis with the navigational clickstream data. Int. J. Electron. Commer. **23**(2), 179–211 (2019)
10. Jindal, H., Sardana, N., Mehta, R.: Analysis and visualization of user navigations on web. In: Hemanth, J., Bhatia, M., Geman, O. (eds.) Data Visualization and Knowledge Engineering. LNDECT, vol. 32, pp. 195–221. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-25797-2_9
11. Scholtes, I.: When is a network a network? Multi-order graphical model selection in pathways and temporal networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, vol. F1296(i), pp. 1037–1046 (2017). https://doi.org/10.1145/3097983.3098145
12. Husin, H.S., Seid, N.: Discovering users navigation of online newspaper using Markov model. In: Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, pp. 1–4 (2017)
13. Wang, G., et al.: Unsupervised clickstream clustering for user behavior analysis. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (2016)
14. Wang, Y.T., Lee, A.J.T.: Mining Web navigation patterns with a path traversal graph. Expert Syst. appl. **38**(6), 7112–7122 (2011)
15. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J.. Comput. **1**(2), 146–160 (1972)
16. Bundy, A., Wallen, L.: Breadth-first search, In: Bundy, A., Wallen, L. (eds.) Catalogue of Artificial Intelligence Tools. Symbolic Computation (Artificial Intelligence), pp. 13–13. Springer, Heidelberg (1984). https://doi.org/10.1007/978-3-642-96868-6_25

17. Barbehenn, M.: A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. IEEE Trans. Comput. **47**(2), 263 (1998)
18. Chakraborty, S., Nagwani, N.K., Dey, L.: Performance comparison of incremental k-means and incremental DBSCAN algorithms. arXiv preprint arXiv:1406.4751 (2014)
19. Barbehenn, M.: A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. IEEE Trans. Comput. **47**2, 263 (1998). https://doi.org/10.1109/12.663776