Received: 30 January 2021 | Revised: 4 July 2021 | Accepted: 15 July 2021
DOI: 10.1002/cpe.6546

WILEY

SPECIAL ISSUE PAPER

Pattern2Vec: Representation of clickstream data sequences for learning user navigational behavior

Erdi Olmezogullari[1] | Mehmet S. Aktas[2]

[1]Development Center, Microsoft, Oslo, Norway

[2]Department of Computer Engineering, Yildiz Technical University, Istanbul, Turkey

Correspondence
Erdi Olmezogullari, Development Center, Microsoft, Oslo, Norway.
Email: erdi.olmezogullari@ozu.edu.tr
Mehmet S. Aktas, Department of Computer Engineering, Yildiz Technical University, Istanbul, Turkey.
Email: aktas@yildiz.edu.tr

Summary

Word embedding approaches represent data sequences to handle their contextual meaning in the NLP tasks. Nowadays, there is an emerging need to understand the user behavior patterns over navigational clickstream data. However, representing the URL data sequences utilizing existing embedding approaches to cluster users' behavior with unsupervised machine learning tasks is a challenging task. This study introduces the Patter2Vec embedding approach using a representation vector to construct contextual, precise, and interpretable clusters over the hidden and popular navigational patterns. To test the usability of the proposed representation in clustering tasks, we conduct an experimental study, which indicates that Pattern2Vec outperforms existing embedding approaches.

KEYWORDS

clickstream, clustering, customer behavior analysis, embeddings, funnel analysis, graph data, user understanding

1 | INTRODUCTION

The navigational clickstream data sequences have hidden information regarding the usability and functionality of web applications. Some traditional sequential pattern mining algorithms (e.g., Prefix, FPGrowth) can uncover popular and hidden user patterns over web clickstream data sequences. However, the evaluation of the patterns is still a challenging problem because of the following two different reasons: 1) The algorithms reveal high volume popular and hidden patterns. 2) The patterns are not even represented with a numerical value easily in unsupervised machine learning tasks. As we proposed, Patter2Vec is creating a highly accurate representation embedding model that addresses those challenges can cluster the representation vector of the navigational user's hidden and popular patterns by leveraging the user clickstream data sequences.

The clickstream records consist of a pile of ambiguous navigational URLs, which do not have any annotated information regarding their actual identity for any analytical tasks. However, we can gather more further general explicit and solid information preliminarily by transforming the form of navigational clickstream data sequences into a web browsing graph because of utilizing vertices and links in the graph, as illustrated in Figure 1. Each link corresponding to the navigational URL between two vertexes shows a single user's action, such as a visited URL. Even though the web browsing graph can visually expose explicit relationships among navigational URLs, a new approach should be needed to summarize user's patterns contextually. We can give two real-world scenarios as an example to explain the navigational graph more correctly, as shown in Figure 1.a1, .a2.

We identify two application use cases, funnel analysis and user interface testing: A funnel analysis is a method to understand the steps necessary to achieve a result on a website.[1] The set of steps is called a "funnel" because the typical shape that visualizes users' flow is similar to a real kitchen funnel. For example, consider an e-commerce company whose ultimate goal is to get users who visit the site to purchase. To this end, the steps necessary to purchase on the site are as follows: visit the site, add a product to the shopping cart, click to validate, and finalize the purchase. These steps are often called conversions. A funnel analysis shows results where users are settling along the conversion path. We can use it to determine where the greatest opportunity is to make improvements.[2] For a funnel analysis to work, it must include all the key steps necessary to achieve a

*Erdi Olmezogullari did this work while he was working in R&D Center at Testinium, Istanbul.

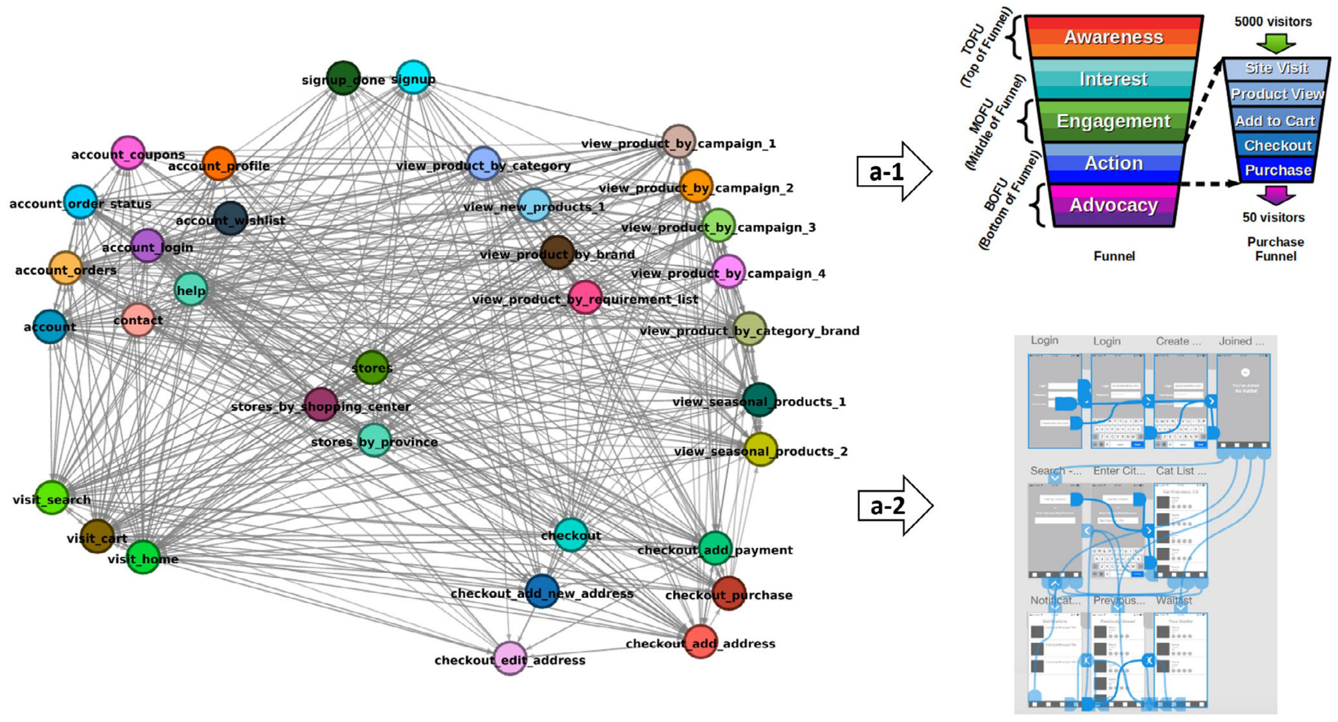


FIGURE 1 Visualization of an example user browsing graph data and application use scenarios (a-1 represents conversion funnel, a-2 represents sequenced GUI test scripts generated for popular sequential actions

conversion. Figure 1 shows an example of funnel analysis graph . Here, the funnel analysis consists of various stages involving awareness, interest, engagement, action, and advocacy. Funnel analytic is a commonly used method to understand better which part of the product experience has the most room for improvement in e-commerce websites. It also involves the "action" stage in which actions are taken to convert a user to purchase a product.[3] We argue that there is an emerging need for methodologies that can automate this by identifying user behavior. In a given funnel analysis, one should always investigate the experience itself and why the users are giving up. However, funnel analysis gives a great starting point so that one can start investigating high-impact areas. To this end, in this study, we investigate a methodology that can provide an automated process for the "action" stage of the funnel analysis.

One of the most challenging processes is to create test codes corresponding to specific test flow scenarios in UI testings. Test analysts and test automation engineers spend most of their time creating the relevant test cases on the website to be tested. Within the scope of the study, we introduce a methodology that can build comprehensive test scenarios automatically. The proposed method automatically finds and classifies popular and hidden patterns by analyzing the user behavior data, that is, clickstream data. Figure 1 illustrates the popular action flow for how they will be created as test flow scenarios. Here, the proposed methodology produces test data sets from the specified patterns obtained from the clickstream data. By utilizing such a test data set, we can generate automated test scripts. To our best knowledge, although there are various test automation tools, there is no tool or methodology that considers both user behavior analysis and machine learning methods.

In the light of the problems as discussed earlier in application use cases, we identify the needs of the Pattern2Vec approach as in the followings: The system should adapt to the changing user browsing behaviors over time when identifying the user behaviors from web browsing graph data. The system should process web browsing graph data, feed the graph data into a machine learning pipeline, and extract user behavior models based on popular sequence data.

In this study, we have the following specific contributions.

- We propose a novel methodology that can capture the user behavior from clickstream data. The proposed methodology identifies clusters of similar user behavior by grouping similar user sequence patterns.
- We perform case studies on real-world clickstream data. We show that our method can capture user browsing behavior categories via unsupervised machine learning by utilizing Pattern2Vec, a sequence pattern-based embedding framework that learns features from user browsing graph data.
- We introduce a methodology for the evaluation of the proposed embedding framework. We use this methodology and compare our approach against the existing embedding frameworks from the literature. The experimental study results indicate that the proposed method can provide a better embedding strategy that can lead to good clustering results and identify unexpected user behaviors.

The structure of this work is as follows. Section 2 gives an overview of the related work, followed by a literature survey. Section 3 describes the problem description. Our proposed methodology is explained in Section 4. Section 5 describes the experimental design, while Section 6 provides empirical results and evaluation of the prototype of Pattern2Vec including data transformations. Section 7 covers the key conclusion related to the summary of the research and future research directions.

2 | LITERATURE REVIEW

In terms of applications, many real relationships are modeled using the best graphical structures. Examples of such applications include social network applications[4,5] internet of things applications,[6] social computing applications[7,8] are just a few examples of real-world structures often modeled by graphs. In this study, we are particularly interested in analyzing the user behavior data on a website by restructuring a web graph at the server logs to get users' browsing behavior.

The embedding approach, a kind of simple neural network, is built on top of two dense hidden layers to create a vector representation for nonnumeric sequential textual documents in natural language processing (NLP) tasks. Each hidden layer calculates an optimum vector weight of the latent features to represent a given text with an n-dimensional vector. For instance, the Word2Vec embedding approach is the most preferred application to produce a numerical vector representation of each word in the sentence in NLP tasks. The Graph embedding methods (e.g., Node2Vec, Graph2Vec, Deepwalk) that are also the most recently used technique are developed on top of the components (e.g., vertex, edge) graph-based dataset. In a graph-based embedding, each graph's component is transformed into a numerical vector representation to use in machine learning tasks. In our context, The sequential user's behavioral actions (e.g., hidden and popular patterns) corresponding to the navigational URLs can be formed as a user behavioral graph to utilize the latent features of graph-based on ML problems. No specific embedding technique in the existing embedding can help represent clickstream patterns with embedding vectors in the literature yet. We proposed Pattern2Vec as a novel methodology that creates embedding vectors corresponding to sequential and clickstream patterns faster and correctly with promising outcomes.

There are also studies that apply unsupervised machine learning approaches to learn from clickstream data.[9-13] However, these studies do not utilize complex user behaviors extracted from the clickstream data.

Some studies focus on distributed collection and management of metadata such as the provenance data[14-18] and information about the web services. In this study, we leave the management of the graph data out of scope. Here, only investigate representation techniques for clickstream data sequences. Yildiz et al. aimed to develop a word2vec word representation by automatically optimizing hyperparameters. In their study, It was observed that the optimization of the values of hyperparameters alone increased classification success significantly.[19,20] Different from this study, we use the word2vec embedding approach as part of our proposed data sequence representation. Our study showed that the proposed pattern2vec representation could improve the clustering accuracy significantly. Some text-based problems (e.g., sentiment analysis, semantic similarity) can use word embedding approaches. For instance, sentiment analysis tasks are conducted by developing a new embedding approach Word2Sent.[21] Short text similarity problem is handled by using a deep learning-based model.[22] Semantic similarity is improved by using Word2Vec on top of two different text-based datasets.[23]

Demirci et al. discuss the use of hierarchical clustering framework for grouping together video streaming systems.[24] Kuwil et al. discuss a clustering algorithm using a critical distance approach.[25] Different from these studies, we use the AgglomarativeClustering hierarchical clustering method and cosine and euclidean distance functions for grouping together data sequences for learning user navigational behavior. There exist studies focusing on LDA-based representation techniques for document clustering.[26] Unlike these studies, we focus on a representation approach representing clickstream-based data sequences for grouping together the data sequences.

3 | PROBLEM DEFINITION

In this manuscript, since we mainly focus on the clustering problem of the patterns collected from web browsing graph data, we developed our novel Pattern2Vec embedding approach to be leveraged in an unsupervised machine learning pipeline. To build such a system, we identify the following research questions:

- What is a pattern-based embedding data representation that includes the important features from web browsing graph data and enables high-quality unsupervised machine learning models to be built?
- What is an efficient algorithm that would utilize such data representation to map a web browsing graph's nodes to a vector?
- What is a good evaluation approach that can demonstrate whether an embedding framework can effectively identify unexpected and complex user browsing behavior?

- Based on such an evaluation approach, how does the performance of the Pattern2Vec embedding framework compare against the benchmark embedding approaches from the literature? How successful is Pattern2Vec in finding the algorithm-generated cluster labels from web browsing graph data?

4 | PROPOSED METHODOLOGY

In this experimental study, to improve the utilization of navigational clickstream patterns in the unsupervised learning tasks, we introduce a new embedding approach by leveraging the Word2Vec embedding approach since the existing embedding approaches are not suitable for representing the patterns. In such a way, before building the Patter2Vec model, we applied many comprehensive consecutive data pipelines, such as data cleansing, data transformation, data mining, and machine learning tasks, to get a relevant dataset. We discussed the core implementation details related to the proposed Pattern2Vec in Section 4.1.

4.1 | Pattern2Vec

In NLP domain, any arbitrary embedding model regarding a text corpus C can be represented by an $N\ x\ d$ dimensional embedding matrix denoted as $EM^C_{N,d}$ in Equation (1). $N\ x\ d$ dimensional embedding matrix has N different embedding vector $\vec{ev}$ that is representing all words in the corpus C with d dimensional latent features. For instance, embedding matrix $EM^C_{N,d}$ is being used as a prebuilt model while aggregating the exact embedding vector for a random sentence S in Equation (2). The sentence S is a kind of collection that consists of random words that an embedding vector can represent as $\vec{EV}^S$. Traditionally, the average function formulated in Equation (3) is applied on each word in the recent existing embedding approaches (e.g., Word2Vec, Deepwalk, Node2Vec). In the average calculation, the sum of vectors is divided by the number of words M in Equation (2). Therefore, the weight of each vector corresponding to the word has equality in terms of contribution while constructing the final embedding vector for sentence S.

$$EM^C_{N,d} = \begin{bmatrix} ev_{1,1} & ev_{1,2} & \dots & ev_{1,d} \\ ev_{2,1} & ev_{2,2} & \dots & ev_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ ev_{N,1} & ev_{N,2} & \dots & ev_{N,d} \end{bmatrix}, \tag{1}$$

$$\vec{EV}^S = \begin{bmatrix} ev(word_1)_{avg} & ev(word_2)_{avg} & \dots & ev(word_d)_{avg} \end{bmatrix}, \tag{2}$$

$$ev(word_j)_{avg} = \frac{\sum_{i=1}^{M} ev_{i,j}}{M}, \tag{3}$$

$$\{word_j | word_j \in C \text{ is a random word in the corpus C and } 0 < j \le M \ll N\}. \tag{4}$$

In contrast to this traditional approach, in this study, we argue that each vector should have different significance, based on their precedence in a sentence, contributing to the final representation of the vector. To do that, we developed a new representation learning technique to adjust the weight of contribution by using navigational clickstream patterns rather than sentences. For instance, a pattern P that consists of u different consecutive navigational clickstream URLs can also be represented with a new embedding model by leveraging the existing embedding models. The pattern P is demonstrated as $P = [url_0, url_1, \dots url_u]$ collected over the user's behavioral actions and fulfills the following insightful facts.

- The pattern P is a unique pattern that can appear many times in the raw data sequences.
- The pattern P is emerged from user browsing graph data by using pattern mining algorithms after performing a couple of sequential data cleansing pipelines.
- The whole combination list, which consists of n-grams of the pattern P does not have any redundant subsequence.
- Any URL $url_x \in P$ can be represented by using our proposal embedding approach Patter2Vec as demonstrated in Equation (5).
- The overall rank of any random URL url_x that appears in the pattern P satisfies Zipf's distribution[27,28] even without sorting URLs by their frequency.

In the Pattern2Vec method, the final aggregated vector $\vec{p2v}^P$ is obtained by including an additional adjustment factor $\vec{a}$, which is adjusting the contribution of Word2Vec embedding vector $\vec{ev}$, as demonstrated in Equations (5) and (6).

We argue that this representation method, which utilizes the Equation (5) is creating the final embedding representation vector $\vec{p2v}^P$ for the pattern P on top of two matrix components, as shown in Equation (7).

$$\vec{p2v}^P = A^P_{1,u} \times EM^P_{u,d}, \tag{5}$$

$$A^P_{1,u} = \vec{a}^P = \begin{bmatrix} a_1 & a_2 & \dots & a_u \end{bmatrix}, \tag{6}$$

$$\vec{p2v}^P = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_u \end{bmatrix}^T \begin{bmatrix} ev_{1,1} & ev_{1,2} & \dots & ev_{1,d} \\ ev_{2,1} & ev_{2,2} & \dots & ev_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ ev_{u,1} & ev_{u,2} & \dots & ev_{u,d} \end{bmatrix}, \tag{7}$$

$$\vec{p2v}^P = \begin{bmatrix} p2v_1 & p2v_2 & \dots & p2v_u \end{bmatrix}, \tag{8}$$

$$a_x = \frac{1}{f_x} \approx \frac{1}{f(k_x)}, \tag{9}$$

$$f(k_x) = \frac{1}{k_x^s \sum_{n=1}^{N}(1/n^s)} \approx \frac{1}{a_x}, \tag{10}$$

$$p2v_{i,j} = \sum_{i=1}^{u} ev_{i,j} \times a_i = \sum_{i=1}^{u} \frac{ev_{i,j}}{f_i}, \tag{11}$$

$$p2v_{i,j} \approx \sum_{i=1}^{p} ev_{i,j} \left[k_i^s \sum_{n=1}^{N} (1/n^s) \right]. \tag{12}$$

The adjustment vector is denoted as $A^P_{1,u}$ where the adjustment a_x term associated with each url_x is constructed by using Zipf's formula together, as demonstrated in Equations (6) and (9). f_x is frequency of url_x in the collection of cleansed sequential clickstream, as stated in Equation (9). We referred to Zipf's law[27,28] to adjust the weight of contribution corresponding to each embedding vector, as presented in Equation (10) since the frequency of all URLs also satisfies a perfect long-tail Zipf's distribution against URLs rank. The rank k_x associated with given a random URL url_x can be calculated by the frequency of URL url_x in Equation (10). Here, k_x shows the rank of URL url_x, and s is a constant used to fit the curve to the exponent characterizing the Zipf distribution. When substituting Equations (11) and (10) into Equation (9), we obtain Equation (12). Therefore, the rank k_i has a direct impact on building the Pattern2Vec vector in terms of the weight of contribution. We used Equation (11) rather than Equation (3) in our proposal Pattern2Vec. Therefore, we introduce a new algorithm, creating a new embedding model to get an embedding vector for the navigational clickstream pattern. The pseudo-code of Patter2Vec is described in Algorithm 1. In the pseudo-code, once the frequency of URL is calculating in line 1. The total number of patterns is identified in line 2. The empty matrices are initialized to store the embedding vector and Patter2Vec vector for each URL in lines 3 and 4, respectively. Each embedding vector corresponding to each URL url_x in pattern P is obtained using the Word2Vec model in line 13. The adjustment a_x factor is applied on each stored ev_x vector between line 16 and 18 as we discussed with Equation (11).

5 | EXPERIMENTAL DESIGN

In the experimental study, we designed experiments on top of three different kinds of components:

- **Platform:** To collect and process clickstream data that belongs to the user's behavior with a highly scalable robust system, we designed and developed a data platform by using distributed systems and cloud services (AWS).
- **Dataset:** We discuss the details of the collected data, used in this study, in Section 5.1.
- **Tools:** As for data processing, we implemented a bunch of applications by using Apache Spark to perform the following tasks: (1) collecting data and transforming JSON to Parquet format, (2) applying data transformations and ML algorithms for Pattern2Vec. To scale the Spark applications we developed, we used AWS EMR Cluster. In addition, We used sckitlearn to apply clustering algorithms and evaluate/visualize the results of clustering. These tasks are explained in Section 5.2.

Algorithm 1 Creating a Pattern2Vec embedding matrix for given patterns, which were generated by sequential pattern mining algorithms

```
Input:
$data$: It is a cleansed sequence dataset.
$patterns$: It consists of the selected patterns
$model$: It is a Word2Vec embedding model that is built with the cleansed sequence.
Output: Creating a Pattern2Vec embedding matrix for given a set of patterns.
Function $Pattern2Vec(data, patterns, model)$ :
1:  $freqs \leftarrow count\_url\_freqs(data)$
2:  $l \leftarrow len(patterns)$
3:  $EM_{l\times d} \leftarrow \{\}[]$
4:  $P2V_{l\times d} \leftarrow [][]$
5:  for $i \in range(0, l)$ do
6:      $P \leftarrow patterns[i]$
7:      $\vec{p2v}^P \leftarrow [0, 0, \cdots, 0]_d$
8:      $u \leftarrow len(P)$
9:      foreach $x \in range(0, u)$ do
10:         $url_x \leftarrow P[x]$
11:         $ev_x \leftarrow \emptyset$
12:         if $url_x \notin \mathrm{EM}_{l\times d}$ then
13:             $ev_x \leftarrow model.transform(url_x)$
14:             $EM_{l\times d}[url_x] \leftarrow ev_x$
            else
15:             $ev_x \leftarrow EM_{l\times d}[url_x]$
            end
16:         $f_x \leftarrow freqs[url_x]$
17:         $a_x \leftarrow \frac{1}{f_x}$
18:         $\vec{p2v}^P \leftarrow \vec{p2v}^P + a_x \times ev_x$
        end
19:     $P2V_{l\times d}[i] \leftarrow \vec{p2v}^P$
    end
20: return $P2V_{l\times d}$
end
```

5.1 A glance to the dataset

5.1.1 Clickstream data

This experimental study collected a raw clickstream dataset over an e-commerce website using a Javascript agent. As Figure 2A shows, we only used three columns out of 26 columns to build data sequences.

SessionId is a unique anonymized identifier column that is a time-based window that contains sequential user's navigational actions. *URL* is navigated to visit a specific page on the website by the user at the specific *timestamp*. *URL* that follows the format <base>/<page> where *base* is the root domain name of a website, *page* that is visited is a page on the website associated with the domain name. We focused on the *page* that has a dynamic shape called *URL* throughout this work.

Figure 2B shows the frequency of the most popular top-10 URLs that were visited via web browsers by users. Here, we obtained this knowledge by applying predefined regular expressions to the dataset. The frequency distribution of URL has a long-tail shape when we plot the whole frequency of URL by their ranks as shown Figure 2C. By analyzing Figure 2, we observe that some of the URL is visited rarely by users. In other words, even though we have high frequently popular visits, such as *view_product* for URL in Table 2, we would also have unpopular URLs with low frequency.

Figure 2C (Zipf's distribution) shows the relationship between frequency and rank of URLs. As shown in this figure, the URL rank increases while the corresponding frequency of URLs is decreasing. This curve can be fitted into a Zipf's distribution as stated in Equation (10) by using maximum likelihood estimation.[29] Therefore, we can formulate our dataset with Zipf's distribution since it has a long-tail shape on the URLs.

Users that can have various persona treat different behaviors while visiting a website during their product purchasing journal. These different behaviors could be limited and generalized by some specific time windows, which are developed over user's visit times:

(A)

Start Date	End Date	Size	Columns
December 2018	March 2019	6.2GB	<SessionId, URL, Timestamp>

(B)

URL	Frequency	Rank
view_product	528655	1
view_product_by_category	381248	2
visit_search	277150	3
visit_home	246828	4
visit_cart	88969	5
view_product_by_campaign_1	67264	6
view_product_by_category_brand	49156	7
account_login	37479	8
account_order_status	31858	9
account_orders	29754	10

(C)

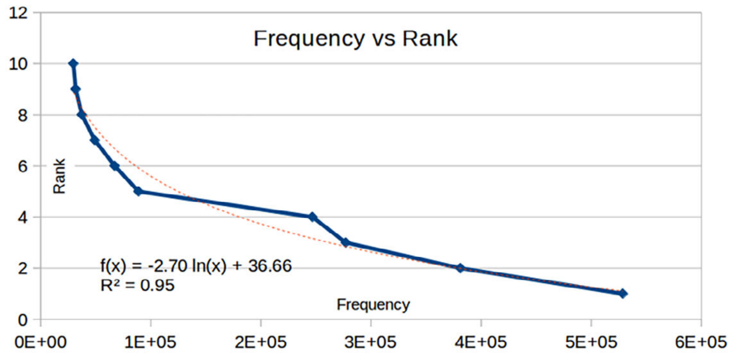


FIGURE 2 (A) Key properties of dataset. (B) Distribution of top 10 URLs by frequency rank. (C) Zipf's distribution

- **Maximum time gap:** The time difference of each arbitrary two consecutive actions that different users treat will not have the same value. It would sometimes be either long or short since the users have different routines during the whole daily activity on the website. We create cumulative distribution functions (CDF) of the time difference as plotted in Figure 3A since we try to identify either valid or outlier user activities accurately. According to this plot, the maximum time gap between two consecutive actions should be strongly (95% of them) less or equal to 3–4 min ($\log_{10}$ 0.5) around.
- **Total maximum session time:** The total duration of a raw session increases due to high time intervals and long event sequence per session. In the raw dataset, 95 of the percentage of the total duration of the raw session consists of is roughly less than 100 min as shown in Figure 3B.

Here, these CDF plots (Figure 3A,B) shed light on user navigational behavior deeply. We set the max time gap at most 3 min to create short sessions rather than long sessions over the raw navigational clickstream dataset. Therefore, the noisy session sequences are trimmed early to get a more relevant dataset for pattern mining before representation learning.

Some critical information about the context of the raw and cleansed dataset, such as the number of an attribute, number of distinct features related, are also presented in tabular (row, column), sequential (session), graph (node, edge) format in Table 1.

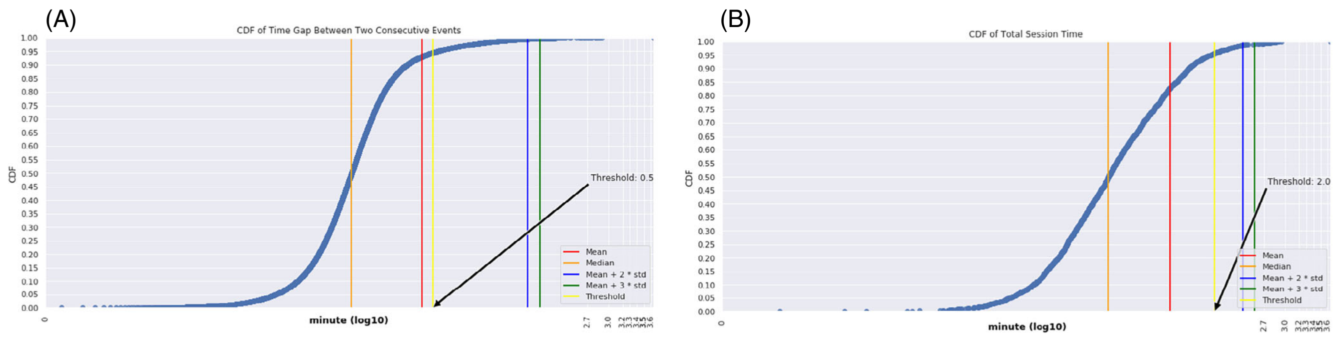


FIGURE 3 (A) CDF of time gaps. (B) CDF of total duration of session

TABLE 1 Key attributes of raw and cleansed navigational clickstream in tabular, sequential, graph format

Data	Explanation	#Row	#Column	#Node	#Edge	#Session	#Url
Raw	Overall	8,615,655	26	16, 836, 952	8, 418, 476	36, 877	16, 836, 952
	Distinct	N/A	26	36, 726	414, 101	1543	36, 726
Cleansed	Overall	813,191	3	2, 232, 428	1, 116, 214	813, 191	2, 232, 428
	Distinct	25,594	3	380	2622	25, 594	380

5.1.2 | Ground truth dataset

The raw navigational dataset does not annotate that we could use to represent any specific group of patterns. We annotated the patterns using the hierarchical clustering and embedding technique after obtaining the popular and hidden patterns over the raw dataset. To get correct clusters for the patterns with a dendrogram, the correlation coefficient C_{P_x,P_y} was computed over each paired pattern $< P_x,\ P_y >$ by applying the below formulas as denoted respectively in Equations (13), (14), and (15). The correlation function is taking two embedding vectors corresponding to the last elements of patterns $EV^{\vec{P}_x[m]}$ and $EV^{\vec{P}_y[n]}$, respectively.

$$P_x = [url_i^x, url_1^x, \dots, url_m^x] \Rightarrow P_x[m] = url_m^x, \tag{13}$$

$$P_y = [url_j^y, url_1^y, \dots, url_n^y] \Rightarrow P_y[n] = url_n^y, \tag{14}$$

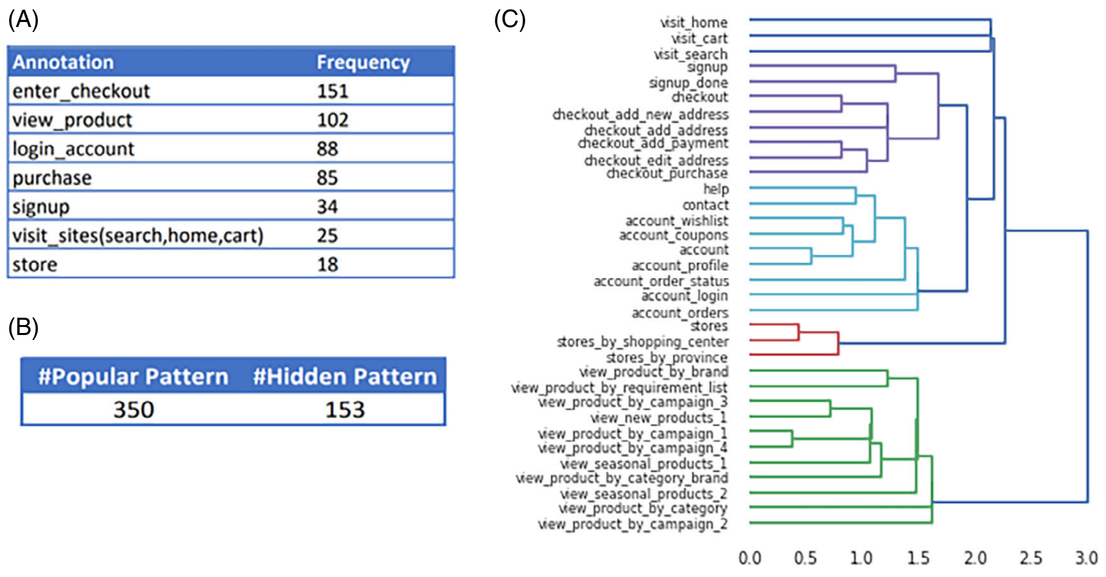
$$c_{P_y,P_x} = c_{P_x,P_y} = corr(EV^{\vec{P}_x[m]}, EV^{\vec{P}_y[n]}), \tag{15}$$

$$CM = \begin{bmatrix} 1 & c_{1,2} & \dots & c_{1,N} \\ c_{2,1} & 1 & \dots & c_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N,1} & c_{N,2} & \dots & 1 \end{bmatrix}. \tag{16}$$

After creating the correlation matrix *CM* by using each correlation coefficient, we get the final annotated patterns that we will use in the evaluation of clustering performance were identified over the dendrogram, as illustrated in Figure 4C. We manually classified the famous and hidden patterns under seven main groups (enter_checkout, view_product, login_account, purchase, signup, visit_sites, store) as shown in Figure 4A,B. We identified either popular or hidden patterns by using the Trie structure in the other data pipeline.

5.2 | Pipelines

To fulfill our proposed Pattern2Vec approach, we developed a bunch of data cleansing and transformations on the raw clickstream dataset toward getting cleansed dataset. In these pipeline implementations, each data transformation stage S_x is consuming data d_x, as summarized in Figures 5, 7, and 8.


(A)

Annotation	Frequency
enter_checkout	151
view_product	102
login_account	88
purchase	85
signup	34
visit_sites(search,home,cart)	25
store	18

(B)

#Popular Pattern	#Hidden Pattern
350	153

FIGURE 4 (A) Frequency of labels. (B) Number of popular and hidden patterns. (C) Hierarchy clustering of correlation matrix *CM*

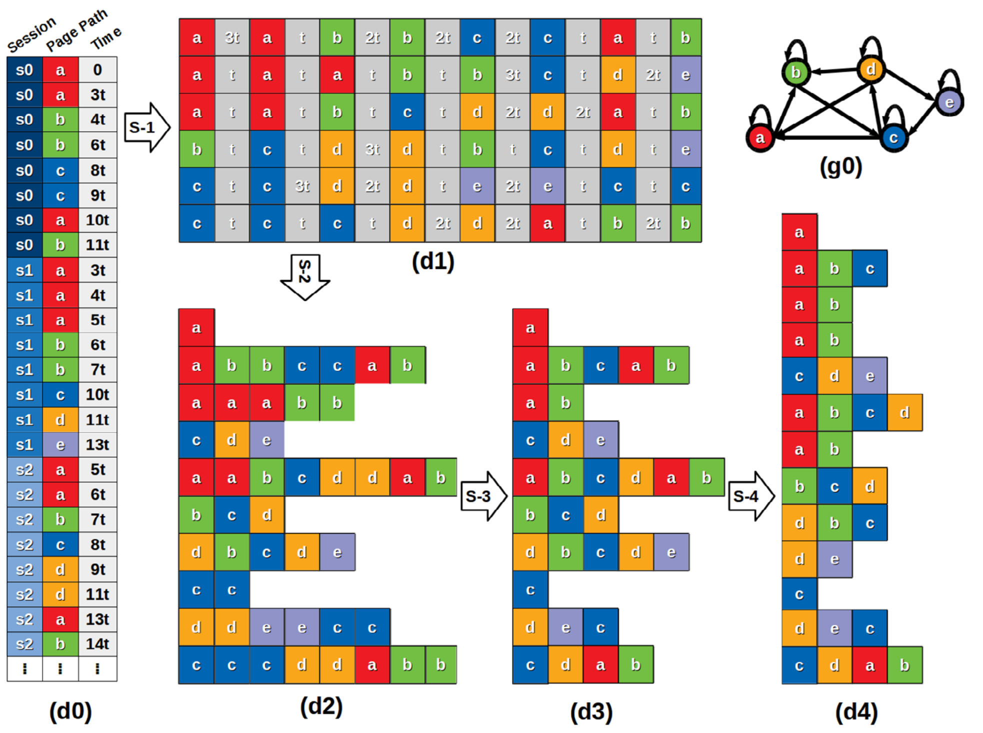


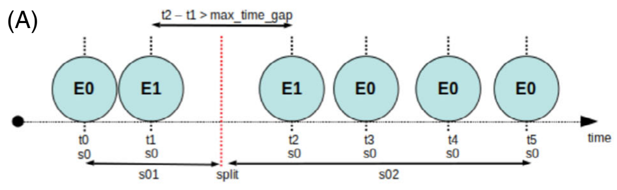
FIGURE 5 Visualization of creating cleansed sequence for *Pattern2Vec*

1. **Data cleansing and correction:** Data formatting and cleansing operations (filtering, data corrections, data generation) s_1 related to text features were performed on the raw dataset, which is denoted d_0 in Figure 4.
   - The missing data points were removed.
   - The specific regular expression were applied to generalize the context of various dynamic URLs under similar predefined topics, as shown in Table 2a.
   - We filtered out the columns <*SessionId,URL,Timestamp*>.

TABLE 2 (a) Regular expressions (regex) are used to associate dynamic URLs with the predefined topic and (b) application of regex on the example navigational URLs

a)	
Regex	**PredefinedTopic**
/.+\\-c([0-9])+/	view_product_by_category
/(.+\\-p\\-.+)/	view_product
/.+\\-b([0-9])+/	view_product_by_brand
…	…
b)	
Navigation URL	**PredefinedTopic**
/XYZ-p-A-1234/	view_product
/ABC-p-B-456/	view_product
/DEF-c321/	view_product_by_category
/XYZ-bl23/	view_product_by_brand
…	…

2. **Creating cleansed sequence dataset:** The clickstream graph navigational URLs g_0 is transformed into a cleansed data sequences with a bunch of transformations, which are denoted as $[s_2 - s_4]$ in Figure 5: SessionId, URLs, and Timestamp are used to create a raw sequence dataset d_1.
   - Creating new subsessions $d2$ over long sessions by applying the threshold *max_time_gap* as illustrated in Figure 6A.
   - Creating sequence d_3 by dropping redundant consecutive URLs.
   - We trimmed self-looping (acyclic) sequence d_4 in the sequences with a sliding window method. The initialized sliding window method moves from the head to the tail of the sequence until it comes across a URL that already appeared in the sliding window. When that condition happened, the stored subsequence in the sliding window is considered as a new session. Then, a new empty sliding window is initialized for the remaining parts of the sequence starting from the last matched point. At the end of this trimming process, we get the final graph denoted as g_1 in Figure 8.
   - The long sessions were split into a couple of new sessions for *max_time_gap*, which was set for 3 min.
3. **Pattern mining:** The pattern mining algorithms (e.g., PrefixSpan, FPGrowth) can find relevant patterns and generate inevitably redundant user behavior patterns over cleansed data sequences. Since the whole patterns sometimes cannot be interpreted quickly and easily, we developed a methodology to eliminate the redundant patterns at the end of this step. We argue that the proposed Pattern2Vec is solving the lack of pattern representation problem in the clickstream dataset. However, we also say that Pattern2Vec and the other embedding approaches could not handle redundant patterns in the early stages unless the raw dataset has predefined annotations related to the patterns. To this end, the redundant filtering patterns enable a better interpretation of embedding results in terms of accuracy. For this reason, we developed a strategy using two different pattern mining algorithms (PrefixSpan and FPGrowth) together. The application of these two different pattern mining algorithms will yield different results. At the same time, both can create common patterns in case the rules are also considered compound patterns. For instance, in our experimental study, we observed that the patterns that PrefixSpan finds out could also appear in the rules, which FPGrowth generates. Therefore, we dropped the redundant patterns by joining the results (patterns, rules) on the prefix and suffix sequences of patterns and rules as illustrated in Table3a–c, respectively. To do this, the parameters *min_support*, *min_confidence*, *min_length* were used, as shown in Table 3.d.
4. **Pattern identification:** We used a Trie data structure to identify the type of generated pattern (e.g., popular, hidden) since Trie is the best way to search any exact pattern in the extensive dataset quickly. The Trie structure is built on top of dataset d_4 that is a cleansed sequence consists of URLs, which is illustrated in Figure 7A. In the Trie structure, the edge between any two nodes represents an URL. The node can store the frequency of common subsequences that appears in the dataset. For instance, while searching a pattern, the search is started from the root node (top to bottom). At the end of the examination, the algorithm can find the targeted pattern in the Trie. Whenever the targeted pattern does not exist in the Trie, it would be identified as a hidden pattern. In other words, each finding pattern would be considered a popular pattern. In our prototype implementation, it turned out that we have 503 unclassified patterns, which are classified under two groups: 350 popular and 153 hidden patterns. When we assume that we have different P patterns whose average length is L^p_{avg}, we also have S distinct subsequences. The average length of those whole sequences is L^s_{avg}, and the length of the most extended sequence is L^s_{max}. In this scenario, we calculated the time complexities for each Trie operation in Figure 7B.
5. **Pattern2Vec embedding—Zipf's law:** As we discussed in Section 4.1, the embedding vector of a pattern d_9 is computed by using corresponding datasets, which are denoted as s_6 and d_7 in Figure 8. In that implementation, we discussed that the frequency of each URL plays an important role since the ranks of URLs satisfy Zipf's Law because the user's attention related to navigational URLs also follows Zipf's Law while navigating URL in a conversion funnel. The attention of users diminishes along the conversion funnel from top to bottom of the funnel. In other words, the conversion rate of a page declines for each step while the user is approaching a targeted page at the conversion funnel. The conversion rate of the two-direction transition that happens between the same two URLs is not the same. Because, the transition of URLs corresponding to an user's rule/pattern (e.g., $url_x \Rightarrow url_y$) does not equal to its symmetric version (e.g., $url_y \Rightarrow url_x$). Each rule/pattern has to be completely different from each other semantically. We argue that the position of visited URLs has a positive or negative impact on its vector representation.



(B)

```
SELECT COLLECT_LIST(page_path) AS raw_sequence
FROM Events
GROUP BY session_id
ORDER BY time ASC
```

FIGURE 6 (A) Creating sessions with *max_time_gap*, which is identified over CDF plots as shown in Figure 3. (B) Creating a raw sequence by using SessionId, URL, and Timestamp

TABLE 3 (a) PrefixSpan result, (b) FPGrowth result, (c) joined results of PrefixSpan and FPGrowth, and (d) pattern mining algorithms and parameters

a)

Pattern	Prefix	Suffix	Support
a, b, c	ab	c	s_0^{spm}
b, c	b	c	s_1^{spm}
a, b, d	ab	d	s_2^{spm}
…	…	…	…

b)

Rule	Prefix	Suffix	Support	Confidence	Lift
b, a ? c	ab	c	s_0^{arm}	c_0	l_0
a, b ? d	ab	d	s_1^{arm}	c_1	l_1
…	…	…	…	…	…

c)

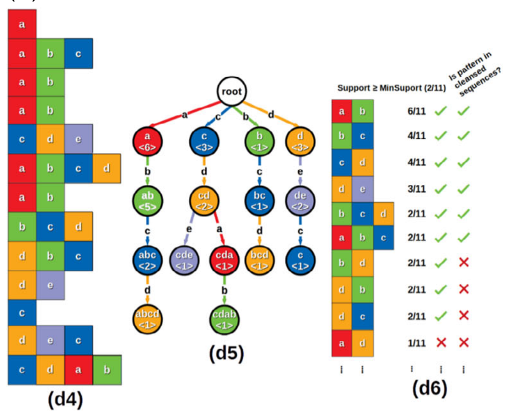
Pattern	Support	Confidence	Lift	Redundant?
a, b, c	s_0^{arm}	c_0	l_0	No
b, c	*Null*	*Null*	*Null*	Yes
a, b, d	s_1^{arm}	c_1	l_1	No
…	…	…	…	…

d)

Algorithm	min_support	min_confidence	max_length[b]
PrefixSpan[a]	0.001	N/A	15
FPGrowth[a]	0.001	0.001	N/A

[a]Apache Spark implementation in Scala.
[b]Determined over cleansed sequence dataset.

(A)



(B)

Operation	Time Complexity
Create	$O(L^s_{max} * S)$
Lookup	$O(L^p_{avg} * P)$
Insert	$O(L^s_{avg} * S)$

FIGURE 7 (A) Building a Trie structure. (B) Time complexity of Trie operation for worst case

Therefore, we developed a novel Pattern2Vec method to include the contribution of the positional information of URL into the vector representation by using Zipf's Law.

6. **Unsupervised learning—Clustering of patterns:** Pattern2Vec approach is evaluated by different clustering methods. The quality of clustering results is demonstrated by using v-score, completeness, homogeneity, silhouette metrics in Section 6. In this pipeline, the clustering of patterns was performed on the different existing embedding models, including Pattern2Vec, as shown in Table 4a.

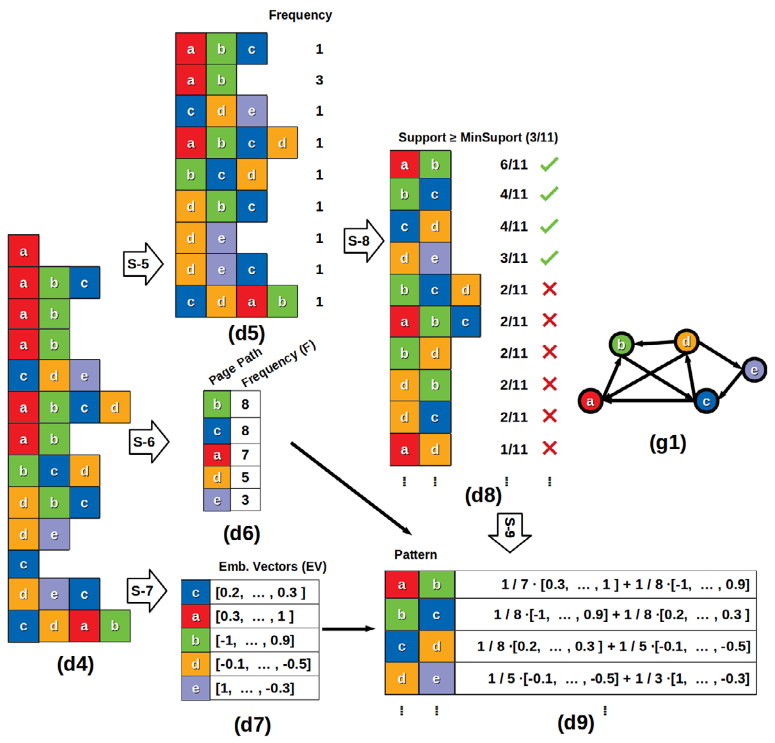


FIGURE 8 Visualization of creating Pattern2Vec embedding vector

TABLE 4 (a) Embedding algorithms and input and output dataset and (b) clustering algorithms and parameters

a)			
Embedding	**Train dataset**	**Test dataset**[a]	
Pattern2Vec[b,c,d]	Cleansed sequence	Cleansed patterns	
Word2Vec[b,c,e]	Raw sequence	Cleansed patterns	
Deepwalk[b,e,f]	Raw graph browsing[g]	Cleansed patterns	
Node2Vec[b,e,f]	Raw graph browsing[g]	Cleansed patterns	
b)			
Clustering name	**Distance**	**Affinity**	**Linkage**
KMeans-1[b,c,h]	Cosine	N/A	N/A
KMeans-2[b,c,h]	Euclidean	N/A	N/A
BiSectingKMeans-1[b,c,h]	Cosine	N/A	N/A
BiSectingKMeans-2[b,c,h]	Euclidean	N/A	N/A
GaussianMixture[b,c,h]	N/A	N/A	N/A
KMeans[b,c,f]	Euclidean	N/A	N/A
AgglomarativeClustering-1[f,h]	N/A	Euclidean	Average
AgglomarativeClustering-2[f,h]	N/A	Euclidean	Ward
AgglomarativeClustering-3[f,h]	N/A	Euclidean	Complete
AgglomarativeClustering-4[f,h]	N/A	Cosine	Complete
SpectralClustring[e,f,h]	N/A	N/A	N/A

[a]Pattern mining parameters in Table 3.d are used.
[b]Number of iteration is 1000.
[c]Apache Spark implementation in Scala.
[d]Additional parameters: max_time_gap = 3 min.
[e]Default parameters are demonstrated in Table 5.
[f]Sckitlearn implementation in Python.
[g]Edges and nodes generated by NGram (k = 2).
[h]Number of cluster is tending between 2 and 25.

6 | EXPERIMENTS

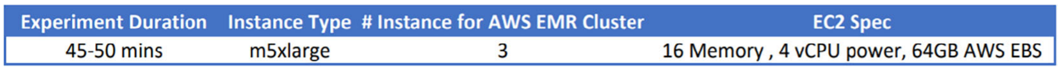
The whole data transformation and data pipelines that were explained under Section 5.2 were run on the Apache Spark Cluster, which is established on three AWS EC instances using AWS EMR. Its type and spec are shown in Figure 9.

6.1 | Experimental study results

We conducted the experiments to prove the quality and performance of Pattern2Vec methodology against the existing embedding approaches (Word2Vec, Deepwalk, Node2Vec) over the generated popular and hidden patterns. To do this,

- We created an embedding model for each embedding method, including Pattern2Vec.
- We applied different clustering techniques on the vector representations corresponding to the patterns, as illustrated in Table 4b.
- Some entropy-based clustering metrics (v-score, homogeneity, completeness) are used to get highly accurate results while calculating over clustered data (predicted labels) and ground truth datasets (actual labels).
- To prove the interpretability of the clustering results with visual evidence, we performed the T-SNE algorithm. Therefore, interpretability and accuracy of Pattern2Vec are comprehensively compared each other existing embedding approaches, as plotted in Figures 10 and 11.

The Pattern2Vec has outstanding results at the whole clustering methods we performed than the rest of existing embeddings, overall. The highest v-score belongs to K-Means-1 that can group the 95% percentage of annotated patterns correctly for six different clusters ($K_{predicted} = 6$), as shown in Figure 10 because it also has the highest homogeneous and completeness scores on the Patter2Vec representation vectors. In consequence of the highest scores, the results of K-Means-1 is a good candidate that can be analyzed more can show the context and shape of the clusters by using T-SNE components on the 2D space. According to Figures 10 and 11, we can see the visual context and shape corresponding to the cluster that supports clearly the clustering metrics' results. In the Figure 11A,B where the first column represents the annotations of patterns, and the rest of each column represent interpretability of K-Means-1 against the different embedding approaches. According to these T-SNE



Experiment Duration	Instance Type	# Instance for AWS EMR Cluster	EC2 Spec
45-50 mins	m5xlarge	3	16 Memory , 4 vCPU power, 64GB AWS EBS

FIGURE 9 Experiment design

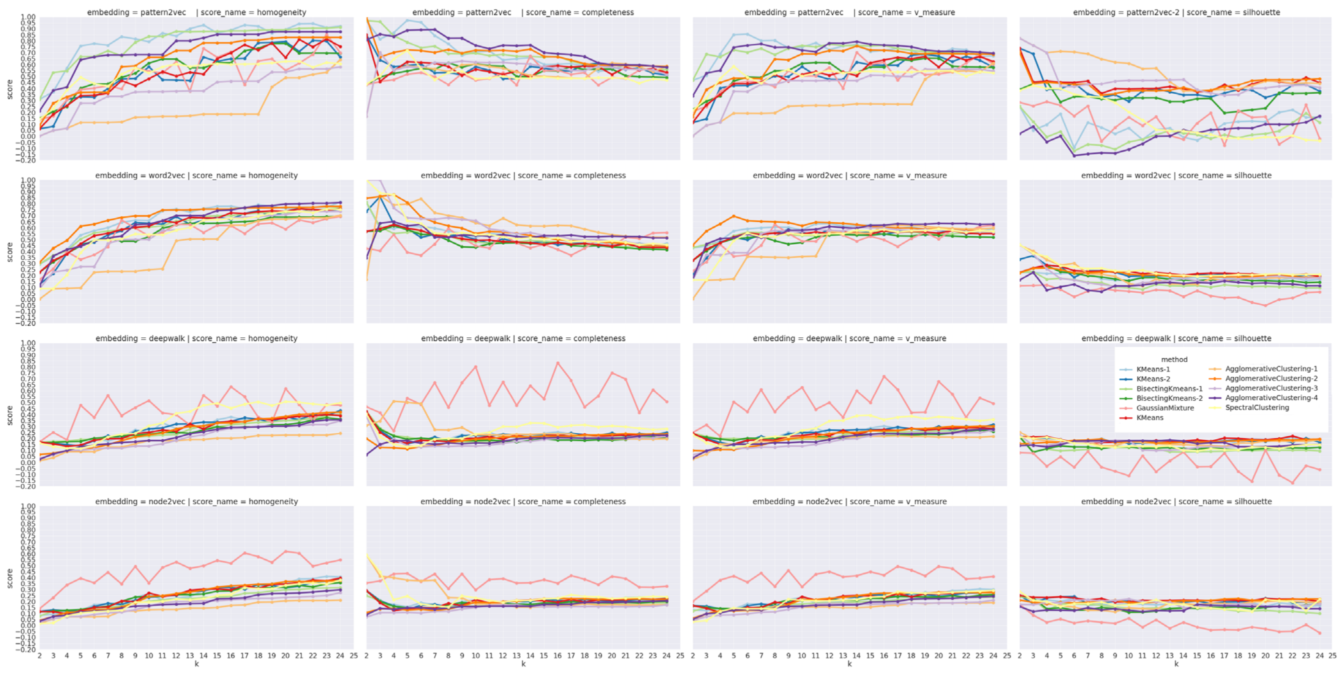


FIGURE 10 Clustering results on for different embedding representations

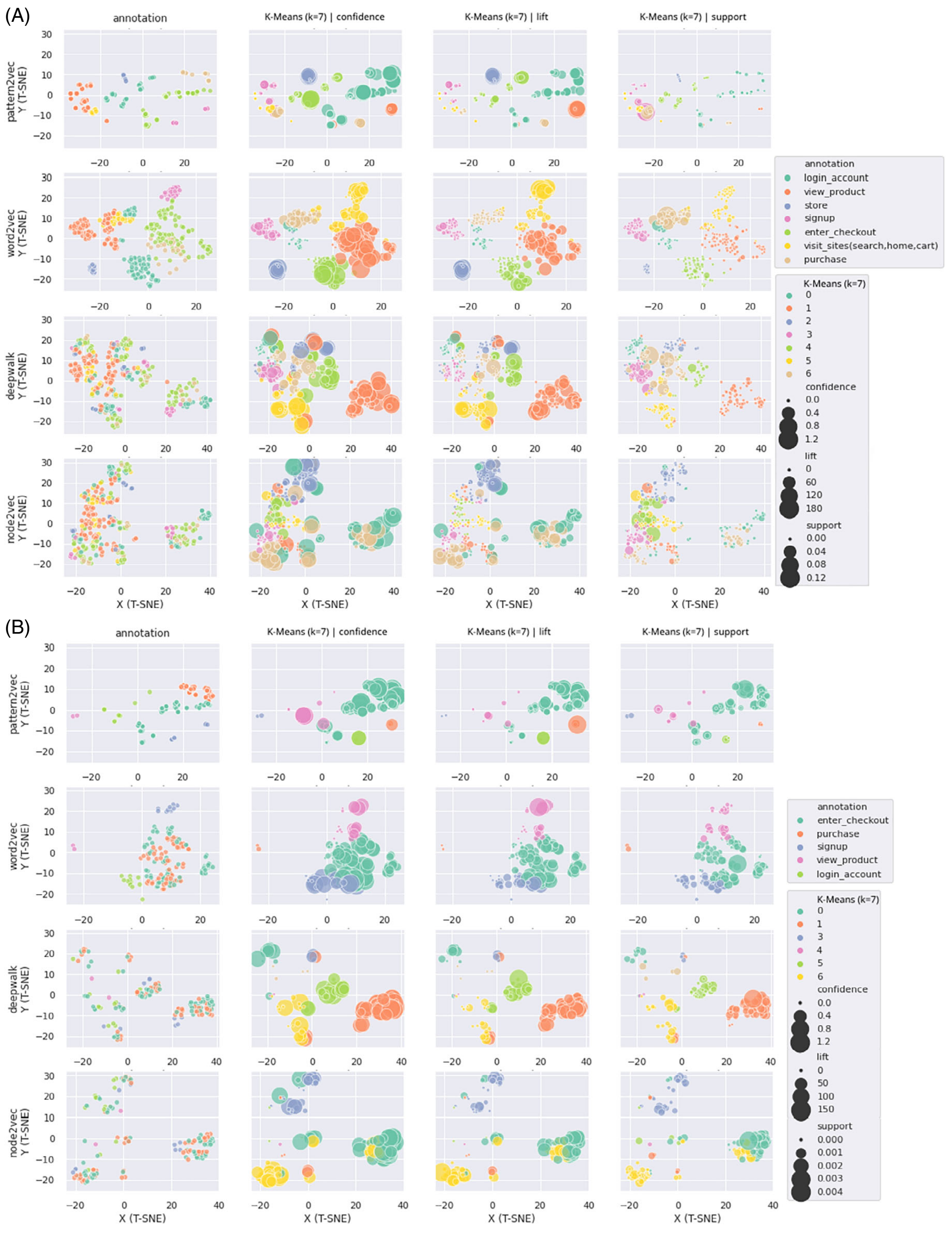


FIGURE 11 Visualization of popular (A) and hidden (B) patterns clustered by KMeans(k=7) with T-SNE

results, the Pattern2Vec approach leads to the best completeness and homogeneous shapes for the clustering results. These results indicate that the interpretability of the cluster can be measurable for our novel Pattern2Vec, which is fulfilling a remarkable result. These outperforming results are outcomes of the adjustment factor, $1/f_x$ shown in Equation (9). It seems that it has a positive impact on the completeness and the homogeneity of a cluster.

In addition to interpretability of those metrics, the traditional evaluation metrics (intracluster, intercluster) can be referred to explain the shapes of the clusters through the grid plots as shown in Figure 11A,B. Here, we observe that the distances of the intracluster are minimized, and our

Pattern2Vec approach maximizes the distances of the intercluster. We argue that this is another result that shows how the clustering of Pattern2Vec vectors is working successfully while identifying the clusters separably. The clusters of the popular and hidden patterns have roughly same centroid origin that can be overlapped on the same 2D space, as shown separately in Figure 11A,B.

The pattern metrics (support, confidence, lift) are added to provide additional information against T-SNE components for the interpretability of embedding approaches, including Pattern2Vec. The sizable data points represent the importance level of predicted and annotated patterns which

TABLE 5 Default parameters of embedding algorithms

Embedding	Parameters	Value
Pattern2Vec	**regex**[a]	See Table 3
	max_time_gap[a]	180000
	min_support[a]	0.001
	min_confidence[a]	0.001
	min_pattern_1en[a]	2
	max_pattern_len[a]	20
	max_iter	100
	vector_size	16
	max_sentence_length	20
	step_size	0.025
	window_size	5
	min_count	5
Word2Vec	max_iter	100
	vector_size	16
	max_sentence_length	20
	step_size	0.025
	window_size	5
	min_count	5
Deepwalk	format	Edge-list
	number-walks	1000
	representation-size	16
	walk-length	15
	window-size	5
Node2Vec	format	Edge-list
	p	1
	q	1
	weighted	False
	directed	True
	window-size	5
	dimensions	16
	walk_length	20
	num_walks	200

[a]These parameters that belong to the core Pattern2Vec implementation are also applied the rest of embedding approaches with an additional implementations to make a fair comparison in this study.

can shed light on the weakness/strongness of the relationship between the targeted approach. The popular patterns roughly satisfy along the x-axis, T-SNE$_x$ on the plots, belonging to Pattern2Vec and Word2Vec:

- The high confidence and lift values are accumulated at the right side of plots between −10 and 30.
- Unlikely, the low confidence and lift values are accumulated at the left side of plots between −30 and −10. For instance, these valuable insights show that the metric confidence and lift patterns may be modeled over embedding vectors by different approaches.

6.2 | Threats to validity of experimental study

This study is conducted on a real-world raw clickstream dataset collected over an e-commerce website between December 2018 and March 2019. In data generalization, we applied a couple of well-defined regular expressions to create a graph browsing dataset. Time-wise feature, new subsequences were produced over the sequence dataset accordingly to the user's navigational behaviors.

We applied a couple of unsupervised techniques on representation matrices for default parameter settings after getting a cleansed proper dataset. In the evaluation step, we developed a specific strategy for Zipf's law to compose our ground truth dataset by annotating the pattern dataset that we obtained over the raw clickstream dataset since it has no labels related to the raw clickstream dataset. The metrics except silhouette are performed on the ground truth dataset. The redundant patterns were cleaned by our proposed experimental method, joining sequential patterns and rules. To establish a proper setup for the comparison of embedding approaches in this study, we also applied the parameters of the core Pattern2Vec algorithm denoted *a* in Table 5 on the rest of the approaches. We managed the scalability of pipelines on top of Apache Spark with a specific cluster on AWS EMR. Under these circumstances, therefore, we accept that any other proper dataset may yield different unexpected results than the results we obtained in this study.

7 | CONCLUSION AND FUTURE RESEARCH DIRECTIONS

This study proposes a representation methodology, Pattern2Vec, for representing user's navigational patterns. Pattern2Vec captures the user behavior from clickstream data. It is a sequence pattern-based embedding framework that learns features from user browsing graph data. We utilize Pattern2Vec for two different application use scenarios: Funnel Analysis and User Interface Testing.

We conducted an experimental study to investigate the usability of the Pattern2Vec in unsupervised machine learning tasks such as clustering and T-SNE tasks. The empirical study results indicate that Pattern2Vec shows better performance against the existing embedding approaches such as Word2Vec, Deepwalk, and Node2Vec for representing the navigational user patterns. Our findings suggest that unsupervised machine learning tasks can use Pattern2Vec to create more accurate and contextual results.

We will design a ranking strategy to query the relevant popular and hidden patterns using the Pattern2Vec approach in future work. We also plan to evaluate the performance of the Pattern2Vec as a unique annotated data representation of the URL data sequences for sequence generation tasks.

ACKNOWLEDGMENT

The authors thank TUBITAK for supporting this research (under project id: 3191534) and Saha Bilgi Teknolojileri Egitim Danismanlık San. Tic. A.S. R&D Center for providing the computational facilities that made this study possible.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Erdi Olmezogullari https://orcid.org/0000-0001-8476-1154
Mehmet S. Aktas https://orcid.org/0000-0001-7908-5067

REFERENCES

1. Mah T, Hoek H, Li Y. Funnel report mining for the MSN network. Paper presented at: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2001:450-455. San Francisco, CA, USA
2. Zhou Y, Mishra S, Gligorijevic J, Bhatia T, Bhamidipati N. Understanding consumer journey using attention based recurrent neural networks. Paper presented at: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2019:3102-3111, Anchorage AK USA
3. Bagherjeiran A, Hatch A, Ratnaparkhi A. Ranking for the Conversion Funnel. Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval; 2010:146-153.

4. Baeth M, Aktas M. An approach to custom privacy policy violation detection problems using big social provenance data. *Concurr Comp-Pract E*. 2018;30(21):e4690.
5. Baeth M, Aktas M. Detecting misinformation in social networks using provenance data. *Concurr Comp-Pract E*. 2019;31(3):85–89.
6. Aktas M, Astekin M. Provenance aware run-time verification of things for self-healing internet of things applications. *Concurr Comp-Pract E*. 2019;31:e4263. https://doi.org/10.1002/cpe.4263
7. Riveni M, Nguyen T, Aktas M, Dustdar S. Application of provenance in social computing: a case study. *Concurr Comp-Pract E*. 2019;31(3):e4894.
8. Olmezogullari E, Ari I. Online association rule mining over fast data. Paper presented at: Proceedings of the 2013 IEEE International Congress on Big Data; 2013. Santa Clara, CA, USA
9. Ruijuan Z, Jing C, Mingcuan Z, Junlong Z, Qingtaoa W. User abnormal behavior analysis based on neural network clustering. *J China Univ Posts Telecommun*. 2016;23(3):29-36.
10. Lin L, Li Y, Ma L. Research on clustering analysis of new energy charging user behavior based on spark. Paper presented at: Proceedings of the 2018 International Conference on Sensor Networks and Signal Processing (SNSP); 2018. Xi'an, China.
11. Ranti K. Clustering stream user behavior data using K-prototypes algorithm. *J Phys Conf Ser Int Conf Eng Technol Innov Res*. 2019;1–8.
12. Wang G, Zhang X, Tang S, Zheng H, Zhao B. Unsupervised clickstream clustering for user behavior analysis. Paper presented at: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16); 2016:225-236. San Jose, California, USA.
13. Hsu J, Yang H. A modified k-means algorithm for sequence clustering. Paper presented at: Proceedings of the 2009 9th International Conference on Hybrid Intelligent Systems; 2009. Shenyang, China.
14. Tufek A, Gurbuz A, Ekuklu OF, Aktas MS. Provenance collection platform for the weather research and forecasting model. Paper presented at: Proceedings of the 2018 14th International Conference on Semantics, Knowledge and Grids; 2018:17-24; Guangzhou, China: IEEE.
15. Tas Y, Baeth MJ, Aktas MS. An approach to standalone provenance systems for big social provenance data. Paper presented at: Proceedings of the 2016 12th International Conference on Semantics, Knowledge and Grids (SKG); August 15, 2016:9-16; Beijing, China: IEEE.
16. Aktas M, Fox GC, Pierce M. Fault tolerant high performance Information services for dynamic collections of grid and web services. *Future Gener Comput Syst*. 2007;23(3):317-337.
17. Olmezogullari E, Aktas M. Representation of click-stream datasequences for learning user navigational behavior by using embeddings. Paper presented at: Proceedings of the 2020 IEEE International Conference on Big Data; 2020. Atlanta, Georgia, USA.
18. Uygun Y, Oguz R, Olmezogullari E, Aktas M. On the large-scale graph data processing for user interface testing in big data science projects. Paper presented at: Proceedings of the 2020 IEEE International Conference on Big Data (Big Data); 2020. Atlanta, Georgia, USA.
19. Yildiz B, Tezgider M. Improving word embedding quality with innovative automated approaches to hyperparameters. *CCPE J*. 2021;e6091. https://doi.org/10.1002/cpe.6091
20. Yildiz B, Tezgider M. Learning quality improved word embedding with assessment of hyperparameters. Paper presented at: Proceedings of the Parallel Processing Workshops. Euro-Par 2019. Lecture Notes in Computer Science; 2020; 11997; Springer, Cham.
21. Kasri M, Birjali M, Beni-Hssane A. Word2Sent: a new learning sentiment-embedding model with low dimension for sentence level sentiment classification. *Concurr Comput Pract E*. 2021.
22. Han M, Zhang X, Yuan X, Jiang J, Yun W, Gao C. A survey on the techniques, applications, and performance of short text semantic similarity. *Concurr Comput Pract E*. 2021.33(5):e5971.
23. Yan F, Fan Q, Lu M. Improving semantic similarity retrieval with word embeddings. *Concurr Comput Pract E*. 2018.30(23):e4489
24. Demirci S, Yardimci A, Sayit M, Tunali ET, Bulut H. A hierarchical P2P clustering framework for video streaming systems. *Comput Stand Interf*. 2017;49:44-58.
25. Kuwil F, Shaar F, Topcu AE, Murtagh F. A new data clustering algorithm based on critical distance methodology. *Expert Syst Appl*. 2019;129:296-310.
26. Onan A, Bulut H, Korukoglu S. An improved ant algorithm with LDA-based representation for text document clustering. *J Inf Sci*. 2017;43:275-292.
27. Adamic LA, Huberman BA. Zipf's law and the internet. *Glottometrics*. 2002;3(1):143-150.
28. Newman M. Power laws, Pareto distributions and Zipf's law. *Contemp Phys*. 2005;46(5):323-351. https://doi.org/10.1080/00107510500052444
29. Wikipedia Maximum likelihood estimation — Wikipedia, the free encyclopedia; 2020. Accessed July 15, 2020. http://en.wikipedia.org/w/index.php?title=Maximum%20likelihood%20estimation&oldid=964549142

How to cite this article: Olmezogullari E, Aktas MS. Pattern2Vec: Representation of clickstream data sequences for learning user navigational behavior. *Concurrency Computat Pract Exper*. 2022;34(9):e6546. doi: 10.1002/cpe.6546