Software: Practice and Experience

WILEY

RESEARCH ARTICLE

An Event Frequency-Inverse Session Frequency Based Weighting Mechanism for Data Embedding Methodologies

Nail Tasgetiren[1] | Ilgin Safak[2] | Mehmet S. Aktas[3]

[1]R&D Center, Hepsiburada, Istanbul, Turkey | [2]R&D Center, Fibabanka, Istanbul, Turkey | [3]Computer Engineering Department, Yildiz Technical University, Istanbul, Turkey

Correspondence: Nail Tasgetiren (nail.tasgetiren@std.yildiz.edu.tr)

Received: 11 April 2024 | **Revised:** 7 December 2024 | **Accepted:** 3 January 2025

Funding: The authors received no specific funding for this work.

Keywords: clickstream data | Deepwalk | E-commerce | embedding | frequency-based weighting | LSTM autoencoder | Node2vec | Struc2vec | user navigational behavior | Word2vec

ABSTRACT

Context: With the rapid increase in data complexity and volume, analyzing and understanding large-scale datasets has become a critical challenge, especially in domains like e-commerce. This paper addresses this challenge by introducing a novel business process that utilizes an event frequency-inverse session frequency (EF-ISF) based weighting mechanism for embedding user navigational behavior in e-commerce websites.

Objective: The primary research problem tackled in this study is the detection of patterns and distributions within user behavior data, which is often complicated by dynamic page content and session sparsity. We aim to enhance the interpretability and accuracy of user behavior analysis by integrating EF-ISF with various embedding techniques, including Word2Vec, Node2Vec, DeepWalk, Struc2Vec, and LSTM Autoencoder.

Methodology: Our approach is particularly effective in datasets where page refreshes lead to event repetitions, where sequential events with temporal dependencies are prevalent, and where event distributions across sessions are sparse. By applying this methodology across multiple e-commerce datasets, we observe consistent performance in clustering quality, successfully detecting underlying patterns and groups within the data.

Results: A prototype implementation demonstrates the practical utility of the proposed method, with evaluation metrics confirming that EF-ISF weighting not only enhances the embeddings but also facilitates a more nuanced and significant segmentation of users' browsing behaviors.

Conclusion: These findings provide a valuable contribution to the field of e-commerce data analytics, offering businesses a refined approach to optimizing user experience and engagement through a deeper understanding of user behavior patterns. This methodology can significantly improve the quality of insights derived from user behavior data, ultimately leading to better decision-making and enhanced user satisfaction.

1 | Introduction

The advent of digital technology has significantly transformed the landscape of data interaction, particularly in the e-commerce sector. The exponential growth in online data, catalyzed by the digitalization of consumer behavior, necessitates advanced methodologies for data analysis and interpretation.

In the context of e-commerce, clickstream data, encapsulating the sequence of clicks made by users, serves as a goldmine for insights into user behavior, preferences, and decision-making processes. Traditional data embedding methods, although effective in capturing structural patterns, often fall short of adequately prioritizing the significance of different elements within the data.

In e-commerce websites, we observe that users often refresh pages frequently, leading to event repetitions. Additionally, some event types are not used by all users, and we see that business processes are triggered when users perform specific sequences of events in a particular order. Defining the data while taking these factors into account becomes a critical requirement for understanding user behavior on e-commerce websites.

This paper delves into the realm of data embedding methodologies, spotlighting the incorporation of an event frequency-inverse session frequency (EF-ISF) [1] based weighting mechanism. This mechanism is pivotal in enhancing the interpretative power and accuracy of data analysis, especially in understanding user navigational behavior on e-commerce platforms. The EF-ISF based weighting mechanism addresses this gap by emphasizing relevant features in the data, thereby refining the embedding process and enhancing the analytical outcomes.

This paper proposes a business process that integrates EF-ISF weighting with embedding techniques such as Word2Vec [2], Node2Vec [3], DeepWalk [4], Struc2Vec [5] and LSTM Autoencoder [6]. This integration aims to offer a nuanced and comprehensive analysis of user navigational patterns, thereby enabling businesses to tailor their strategies and interfaces to better suit customer needs and behaviors. Through the lens of this methodology, we explore the potential of EF-ISF to elevate the granularity and precision of user behavior analysis on e-commerce websites.

Our exploration is underpinned by a prototype implementation that showcases the effectiveness of the proposed methodology. By evaluating the quality of clustering metrics and the interpretability of the results, we demonstrate the enhanced capability of our approach in understanding and interpreting complex user navigational data. This manuscript gives a detailed examination of the literature, methodology, and empirical findings that substantiate the utility of the EF-ISF based weighting mechanism in the domain of data embedding methodologies.

The contributions of this study are as follows:

1. A novel EF-ISF based weighting mechanism for enhancing data embedding methodologies is introduced, which significantly improves the analysis of user navigational behavior on e-commerce platforms.
2. The integration of the EF-ISF weighting mechanism with natural-language-based, graph-based and sequence-to-sequence based embedding methods, namely Word2Vec, Node2Vec, Struc2Vec, DeepWalk and LSTM Autoencoder [6], is proposed. This integration is instrumental in refining the representation of user behavior, leading to more accurate and insightful clustering of user navigational patterns.
3. An empirical evaluation is conducted to validate the effectiveness of the EF-ISF based weighting mechanism. Using clickstream data from two different e-commerce site, the study investigates how the proposed approach performs in data representation compared to traditional embedding methodologies that do not incorporate any weighting mechanism. To do this, the proposed EF-ISF based methodology is examined through partition-based clustering algorithms such as K-Means [7], density-based clustering algorithms like Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [8], and model-based clustering algorithms exemplified by the Gaussian Mixture Model (GMM) [9], showcasing its adaptability and efficiency in analyzing complex user behavior datasets.
4. Empirical verification shows that the proposed EF-ISF weighting mechanism integrated with the sequence-to-sequence (LSTM Autoencoder) data embedding method greatly enhances clustering quality. The results demonstrate that the proposed method effectively captures the nuances of user navigational behavior, providing deeper insights into the underlying patterns and distributions within the data.

This paper is structured as follows: Section 2 outlines the research questions. Section 3 provides an overview of the background work and the literature review. Section 4 presents the methodology. Section 5 describes the implementation details and an evaluation of the proposed weighting methodology. Finally, Section 6 is dedicated to the conclusions and future work.

2 | Research Questions

This paper aims to focus on refining data embedding methodologies for analyzing user navigational behavior on e-commerce platforms, employing an EF-ISF based weighting mechanism.

In this context, the following questions are in order:

Question-1: How can integrating EF-ISF based weighting mechanism with traditional embedding methodologies improve the accuracy and depth of user navigational behavior analysis on e-commerce platforms? Question-2: What are the implications of incorporating EF-ISF weighting on the quality and interpretability of the embeddings generated from user clickstream data? This question focuses on evaluating how the EF-ISF based approach modifies the representation of user sessions in the embedding space, thereby affecting the outcome of clustering and behavior analysis. Question-3: How does the integration of EF-ISF weighting with natural language-based embedding techniques, such as Word2Vec, and graph-based embedding techniques such as Node2Vec, DeepWalk, Struc2Vec, sequence to sequence based embedding technique, such as LSTM Autoencoder influence the clustering performance in segmenting users

based on their navigational patterns? This question investigates the effectiveness of EF-ISF in enhancing the discriminative power of embeddings generated by these methods, leading to more meaningful and actionable clustering of user behaviors. Question-4: How does the EF-ISF weighting mechanism perform when applied to two distinct datasets—one with dynamic page content and frequent user refreshes, and the other characterized by sequential events with temporal dependencies? Specifically, can IF-ISF weighted embeddings generated through NLP-based, graph-based, and sequence-to-sequence approaches achieve similar performance in detecting distributions and clusters using methods like K-Means, DBSCAN, and Gaussian Mixture Model (GMM) across these varying datasets? Question-5: What are the differences in data representation performance between sequence-to-sequence (LSTM Autoencoder) based embedding and graph-based or NLP-based embedding methods?

A careful study of these research questions contributes to the development and validation of a methodological framework that leverages a EF-ISF based weighting mechanism to enhance the precision and utility of data embedding methods for user behavior analysis on e-commerce websites.

3 | Background Work and Literature Review

3.1 | Background

In this study, we explore various embedding approaches that are widely used to represent data in a lower-dimensional space. One of the foundational methods is Word2Vec [10], which predicts a target word based on the context provided by its surrounding words, effectively capturing semantic relationships between words. Word2Vec employs two main algorithms: Skip-Gram [11], which predicts the surrounding words given a target word, and CBOW (Continuous Bag of Words) [12], which predicts a word from its context. The resulting embeddings position semantically similar words close to one another in the vector space, making Word2Vec particularly useful for tasks that involve understanding word meanings in context.

Node2Vec [13] extends the Word2Vec approach to graph-structured data, where the goal is to predict a node's context, or its neighboring nodes, by simulating random walks through the graph. Node2Vec employs biased random walks that balance breadth-first and depth-first searches, capturing both local and global graph structures. The learned node embeddings not only reflect the immediate neighborhood of each node but also capture the overall topology of the graph, making it an effective tool for a wide range of graph-based tasks.

Another method, Struc2Vec [5], focuses on capturing the structural similarity between nodes in a graph rather than their proximity. While Node2Vec emphasizes local neighborhoods, Struc2Vec builds hierarchical representations based on a node's structural identity, such as its degree or centrality. By grouping nodes with similar roles, Struc2Vec ensures that structurally similar nodes, even if distant in the graph, are represented similarly in the embedding space. This makes Struc2Vec particularly useful for tasks where the functional role of a node in a network is more important than its immediate connections.

DeepWalk [4], like Node2Vec, also learns embeddings by simulating random walks [14] on graphs. However, unlike Node2Vec's biased random walks, DeepWalk uses uniform random walks, which focus more on the local neighborhood around each node. DeepWalk treats the generated node sequences as sentences in Word2Vec, applying the Skip-Gram model to learn the embeddings. This approach emphasizes capturing the proximity-based relationships between nodes and is particularly effective for tasks that require an understanding of local graph structures.

The LSTM Autoencoder [15] offers a different approach, designed to handle sequential data such as time series or clickstream events. Its objective is to learn an efficient representation of sequences by encoding and decoding them, capturing complex temporal patterns and dependencies. An LSTM Autoencoder network is used to compress a sequence into a compact representation, which is then reconstructed by another LSTM Autoencoder network. The aim is to minimize the reconstruction error, enabling the model to learn useful patterns within the sequence. This approach is well-suited for tasks that involve sequence clustering, where the temporal aspect of the data is crucial.

There are several key distinctions between these embedding approaches. In terms of prediction objectives, Word2Vec predicts the context of words based on their co-occurrence in sentences, while Node2Vec and DeepWalk focus on predicting the context of nodes in a graph. Node2Vec uses biased random walks to capture both local and global graph structures, whereas DeepWalk employs uniform random walks. Struc2Vec, on the other hand, does not rely on proximity-based predictions but instead focuses on structural similarity, predicting nodes based on their functional role in the graph. The LSTM Autoencoder differs significantly by focusing on sequences, predicting patterns and relationships within sequences of data through compressed representations. The nature of the data handled by these approaches also varies. Word2Vec is specifically designed for linear sequences, such as text, while Node2Vec, Struc2Vec, and DeepWalk are used for graph-structured data. In contrast, the LSTM Autoencoder is tailored for sequential data that contain temporal dependencies, such as time series or clickstream [16] events.

Each of these methods has its own unique strengths. Word2Vec excels at capturing semantic relationships in text data, making it ideal for natural language processing (NLP) tasks. Node2Vec and DeepWalk are both effective at capturing the structures and relationships between nodes in graph data. Struc2Vec stands out for its ability to capture structural roles within a graph, making it suitable for applications where the role of a node in the network is more important than its proximity to other nodes. Finally, the LSTM Autoencoder is particularly useful for modeling temporal and sequential patterns, providing a powerful tool for tasks that require the representation of complex time-based data.

3.2 | Literature Review

In the realm of data analysis, especially within the context of e-commerce, understanding user behavior through clickstream data has become increasingly crucial [17–19]. Traditional embedding methods like natural language-based embedding approaches such as Word2Vec [20–22] and graph embedding

approaches such as Node2Vec [20, 23, 24], Struc2Vec [25–27] and DeepWalk [28–30], have been pivotal in translating raw data into meaningful insights, facilitating user behavior analysis and segmentation [31]. However, these methods often overlook the differential importance of data elements, leading to potential oversimplifications in the analysis.

The use of different weighting mechanisms integrated with data embedding methodologies has emerged as a promising solution to this challenge [32–34]. However, these approaches overlook the importance of data elements relative to whole document collection corpus. Different from previous studies, this study, by applying EF-ISF weighting mechanism to user's browsing data elements, investigate whether it's possible to enhance the discriminate power of the embedding vectors representing the user browsing sessions, thereby providing a more nuanced understanding of user navigational patterns.

NLP based methods refer to techniques originally designed for processing and analyzing textual data, such as words, sentences, and documents. These methods are characterized by their ability to capture semantic relationships between elements in a sequence using embeddings that reflect contextual similarity. In the context of this study, NLP-based methods, such as Word2Vec, are adapted to represent user interactions (e.g., page clicks) in a manner analogous to processing words in a sentence. This adaptation enables the use of proven techniques from NLP to model and analyze sequential clickstream data in e-commerce environments.

Despite the recent advances, the literature reveals a gap in the comprehensive application and evaluation of frequency based weighted embedding methodologies in analyzing clickstream data [35, 36]. To the best of our knowledge, there are no studies that have systematically assessed the impact of frequency based weighting methods across different NLP-based, graph-based, and sequence-to-sequence embedding methods and their effectiveness in clustering user navigational behaviors. This paper seeks to bridge this gap by conducting an in-depth exploration of EF-ISF based weighting mechanisms in conjunction with various embedding techniques, evaluating their efficacy in capturing and analyzing the complexities of user navigational behavior on e-commerce platforms. This paper hence aims to contribute to the evolving landscape of data analytics, offering insights into the development of more sophisticated and accurate methods for understanding user behavior in digital domains.

Recently, researchers have looked into graph embedding methods that use deep learning techniques, mainly LSTM Autoencoders, for a range of purposes. Seo and Lee [37] suggested using LSTM Autoencoders to insert weighted graphs by encoding node-weight sequences into fixed-length vectors. In the same way, Taheri et al. [38] came up with a way to use LSTM Autoencoders to embed graph sequences made from breadth-first search, random walks, and simplest paths. Chu et al. [39] presented DEGREE, an e-commerce-related deep learning-based method that uses both inter-group and intra-group proximities for ranking optimization. Their study showed better matching performance on real-world datasets. Pujastuti et al. [40] solved the sparse rating matrix problem in e-commerce recommender systems by mixing LSTM, Stack Denoising Auto Encoder (SDAE), and latent factor models. These studies show that LSTM-based Autoencoders and graph embedding methods work well in some situations, such as e-commerce and graph classification tasks. However, these approaches have used LSTM Autoencoders for different tasks, like modeling sequential data and predicting time series, most of which only looked at how to use LSTM networks without accounting for the graph structure. LSTM-based designs have only been used with sequential data, and ignored the rich relational information that is stored in graph structures.

Various graph embedding algorithms can be used to obtain information and interpret a graph. One of these is the deep learning-based algorithm LSTM autoencoder [41–46]. In this study, we utilized LSTM autoencoders along with both NLP, Word2Vec and graph embedding techniques, Node2Vec, DeepWalk and Struc2Vec, integrated with the proposed weighting mechanism. Integrated with the aforementioned embedding approaches, the proposed weighting method in this paper can successfully capture both semantic relationships and structural dependencies at the same time.

Graph embedding techniques seek to represent nodes in a graph as vectors while maintaining structural integrity. Recent studies have concentrated on enhancing the efficacy and caliber of these embeddings. GEMSEC concurrently acquires node embeddings and clustering, integrating social network characteristics via regularization [47]. DAOR provides a parameter-free methodology that generates compact, interpretable embeddings resilient across metric spaces [48]. A scalable unsupervised approach utilizing a divergence score has been developed to assess the embedding quality [49]. Existing research on collaborative systems, grid computing, and advanced data processing methodologies forms the foundation for this study. Nacar et al. [50] and Aydin et al. [51] highlighted the scalability and efficiency of grid-based platforms in computational and geographical information systems, respectively, while Aktas et al. [52] demonstrated the integration of computational grids with GIS web services through iSERVO project. Fox et al. [53] emphasized the interplay between computational models and resource optimization within grids. Recent advancements such as Uygun et al. [54] focused on large-scale graph data processing for big data applications, and Olmezogullari and Aktas [55, 56] introduced embedding-based representations for clickstream data to enhance user navigational behavior analysis. Sahinoglu et al. [57] conducted a systematic mapping study on mobile application verification, emphasizing structured analysis for ensuring system robustness, which complements our approach to improving data processing reliability. Furthermore, Pierce et al. [58] addressed web services for managing geophysical data, while Kapdan et al. [59] explored structural code clone detection using software metrics, emphasizing efficient data handling. Building upon these foundational works, our study diverges by introducing a novel EF-ISF based weighting mechanism tailored to e-commerce platforms. Unlike prior studies focusing on infrastructure scalability [50, 51] or graph and metric-based data processing [54, 59], or systematic application verification [57], our work integrates multiple embedding techniques (Word2Vec, Node2Vec, LSTM autoencoder) with EF-ISF to capture sequential and structural patterns in user clickstream data, offering deeper insights into user behavior analysis.

The Davies–Bouldin index (DBI) has been examined as an optimization criterion for clustering [60]. The DBI and Silhouette score are methods typically used to assess the quality of clusters generated by embedding techniques that convert data into a lower-dimensional space to uncover underlying patterns. The Silhouette score quantifies the similarity of an object to its cluster relative to other clusters, reflecting cluster cohesion and separation, whereas the DBI assesses clustering quality by comparing the average similarity ratio of each cluster with its most similar counterpart, with lower values signifying superior clustering.

Traditional weighting mechanisms, such as term frequency-inverse document frequency, focus primarily on textual data and emphasize the relevance of terms within a document relative to a corpus. However, these methods often fail to capture the structural and temporal nuances of sequential datasets like the session-based clickstream datasets. Our proposed EF-ISF mechanism addresses these gaps by weighting events based on their occurrence frequency and their rarity across the entire dataset particularly in the context of graph and sequence-to-sequence embedding spaces and session-based clickstream data domain. This dual perspective ensures that both commonly repeated and contextually significant events are effectively represented in the embedding spaces like graph and sequence-to-sequence embedding spaces. Unlike static weighting schemes, EF-ISF adapts to the nature of the structural data (i.e., linked structure within the navigational browsing graph data) and temporal data (i.e., sequential page visits within the navigational browsing graph data), offering a robust framework for analyzing user's navigational behavior. To the best of our knowledge, no existing approach integrates both event-specific and session-specific weighting to refine embeddings within the context of structural and temporal embedding spaces in the clickstream data domain.

In the realm of document representation and authorship analysis, Agarwal et al. [61] introduced a method that combines TF-IDF weighting with word embeddings to enhance authorship clustering. Their approach effectively captures both semantic and syntactic features, leading to improved clustering performance. Similarly, Schmidt [62] explored various techniques to compute dense document embeddings using word vectors, proposing methods that optimize weighted cosine similarity measures. These studies underscore the significance of integrating term-weighting schemes with embedding techniques to advance document analysis tasks. This study differentiates itself by introducing the EF-ISF weighting mechanism, which uniquely addresses the complexities of user interaction data in e-commerce environments. Unlike traditional term-weighting methods such as TF-IDF, which are predominantly designed for NLP tasks, EF-ISF captures both structural and temporal dependencies within user sessions.

4 | Methodology

In this study, we propose a methodology for clustering users' session-data based on the clickstream data that was obtained by monitoring an e-commerce website. The proposed methodology identifies user behavior and groups users' sessions. The proposed business process is shown in Figure 1. This business process consists of various modules: Embedding function module, weighting function module, weighted average based vectorization module, and clustering module.

Embedding Function Module: In this study, we employ five distinct embedding algorithms to represent user navigational behavior. These include one natural-language-based embedding, Word2Vec, three graph-based embeddings—Node2Vec, DeepWalk, and Struc2Vec—and a sequence-to-sequence model, the LSTM Autoencoder.

The Web log file contains detailed session information about user interactions, tracking every click-event performed during a session, which is identified by a sessionID. The embedding function module plays a role in this process, transforming sessional clickstream data into numerical vectors by utilizing

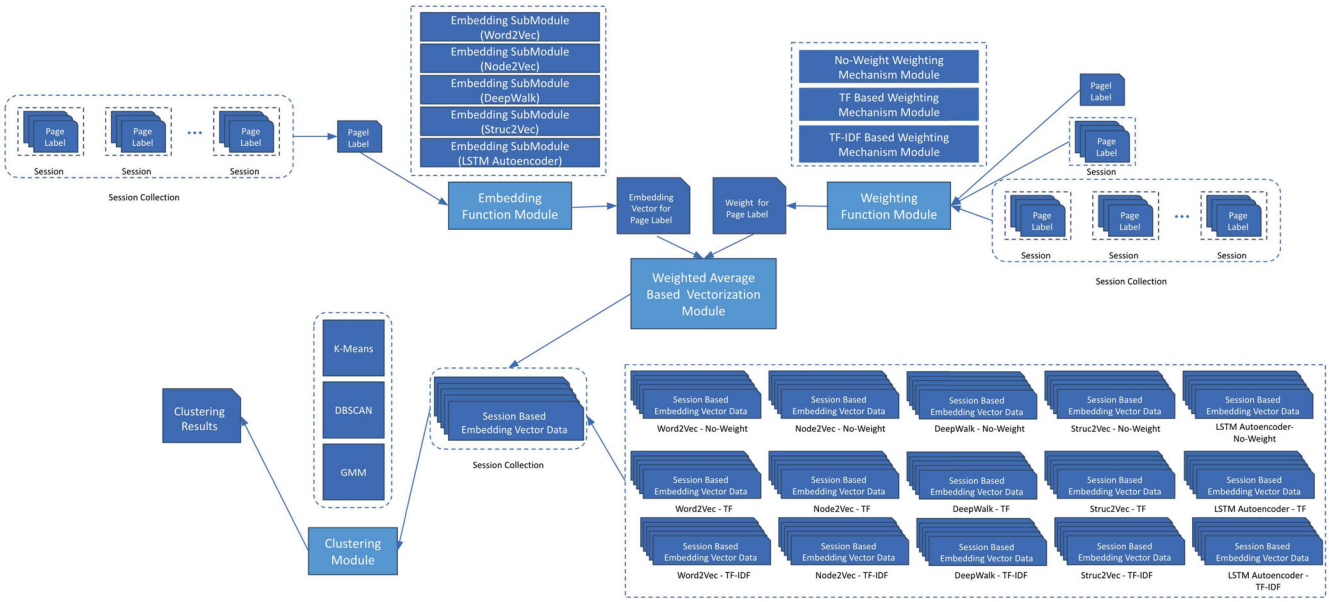

FIGURE 1 | Overview of software architecture for the proposed business process.

various embedding functions. The embedding function module incorporates an embedding function responsible for executing the different embedding methods. This function chooses the appropriate embedding method to convert a page label into a numerical vector based on the embeddingFunctionType, as shown in Algorithm 1.

Word2Vec is a natural-language-based embedding model that generates dense vector representations for words, in this case, page IDs in our dataset, based on their contextual relationships within the clickstream data. The model employs two key architectures: CBOW and Skip-Gram. In this study, we use Skip-Gram, which aims to predict context words surrounding a target word (or page ID). Word2Vec allows us to capture the semantic relationships between web pages by learning vector representations where similar pages are placed closer in the embedding space.

In Word2Vec, each unique token (e.g., a page ID in our dataset) is assigned a corresponding row in an embedding matrix. During training, the model aims to optimize this embedding matrix such that tokens appearing in similar contexts are placed closer together in the vector space. This is achieved by maximizing the cosine similarity (or equivalently, the inner product) between neighboring embeddings and minimizing it for random embeddings. In the CBOW architecture, prediction involves summing the context embeddings of surrounding tokens and using the resultant vector to predict the target token. Conversely, in the Skip-Gram model, the objective is to predict the context embeddings for a given target token. The cosine similarity serves as a measure of alignment between vectors, ensuring that embeddings for related tokens are more similar. By leveraging this approach, Word2Vec effectively captures semantic and contextual relationships within the sequential data, enabling meaningful representation of user navigational behavior.

In this study, NLP-based embedding methods are used to model sequential interactions within user sessions. Specifically, Word2Vec, an NLP-based method, treats page IDs as tokens analogous to words in a text corpus. By capturing the contextual relationships between tokens, Word2Vec generates dense vector representations that reflect the sequential and co-occurrence patterns of user interactions. This approach leverages the principles of NLP to provide meaningful representations of non-textual data, making it particularly effective for clickstream analysis.

DeepWalk is a graph-based embedding method designed to learn representations of nodes in a graph by applying random walks, similar to how Word2Vec learns word embeddings. The algorithm simulates random walks through a graph, treating each walk as a sentence of node sequences. These sequences are then fed into the Word2Vec model to generate embeddings. The strength of DeepWalk lies in its ability to discover latent representations of nodes based on their graph context, revealing both the local and global structure of the graph.

Node2Vec builds upon the concepts introduced in DeepWalk but extends its methodology by incorporating a more nuanced approach. To generate random walks within a graph, it combines Depth-First Search (DFS) and Breadth-First Search (BFS) algorithms. A random walk is a stochastic process in which entities move randomly from one node to another. Steps are taken at each node following random decisions, but not predetermined sequences. Two critical parameters influence the behavior of random walks: P (return parameter) and Q (in-out parameter). A large P value tends to result in a longer random walk since the probability of backtracking the walk is lower, hence facilitating a better exploration of the graph. In contrast, a low value of the parameter P increases the likelihood of backtracking during a random walk, keeping the random walk closer to its starting point. There is a similar, yet opposite pattern of behavior for parameter Q; a small value for Q promotes exploration, whereas a large value for Q promotes more localized random walks. Both the Node2Vec and DeepWalk embedding algorithms require the input of a graph. Node2Vec and DeepWalk embedding algorithms produce a numerical vector as the output.

ALGORITHM 1 | Embedding function.

Input: pageLabel, embeddingFunctionType.
Output: numericalVector.

1. If embeddingFunctionType == “Word2Vec”:
   numericalVector = WordToVec(pageLabel).
2. Else if embeddingFunctionType == “Node2Vec”:
   numericalVector = NodeToVec(pageLabel).
3. Else if embeddingFunctionType == “DeepWalk”:
   numericalVector = DeepWalk(pageLabel).
4. Else if embeddingFunctionType == “Struc2Vec”:
   numericalVector = StructToVec(pageLabel).
5. Else if embeddingFunctionType == “LSTM Autoencoder”:
   numericalVector = LSTMAutoencoder(pageLabel).
6. Else:
   Print “Error: Unknown embedding function type” and exit.
7. Return numericalVector.

The Struc2Vec embedding algorithm endeavors to construct vector representations for nodes in a graph, emphasizing the consideration of the topological characteristics of the graph structure. The process commences by selecting an initial node, acting as the starting point for vector creation. Subsequently, a traversal step is executed, wherein the algorithm explores the graph structure systematically, identifying nodes with similar structural positions through a traversal algorithm. During each traversal step, the algorithm generates embedding vectors for the nodes, incorporating information from the node itself and its neighboring nodes. Notably, the model amalgamates these vectors with those of nodes sharing akin structural connections, thereby accentuating the structural similarities among nodes. The Struc2Vec embedding algorithm requires the input of a graph and produces a numerical vector as the output.

Both Node2Vec and Struc2Vec start by taking node IDs as input and producing embeddings as the output. In Node2Vec, the process involves generating sequences of nodes through biased random walks and then feeding these sequences into a Word2Vec-like embedding process. Struc2Vec, on the other hand, constructs hierarchical representations of structural similarity for the nodes and encodes these into embeddings through iterative aggregation. While both methods transform node IDs

into embeddings, Node2Vec emphasizes proximity-based relationships, and Struc2Vec focuses on structural roles.

The Long Short-Term Memory (LSTM) autoencoder algorithm is designed to model how nodes in a graph and their connections evolve. The LSTM autoencoder does not directly take IDs as input or use a dedicated embedding layer to transform IDs into embeddings, unlike methods like Word2Vec or Node2Vec. In this embedding approach, each node in the graph is initially represented by a set of features, which may include static attributes (e.g., degree, centrality), dynamic interaction metrics (e.g., temporal frequency of connections), or pre-computed token embeddings if available. These features are transformed into a time-series format to capture temporal dependencies and patterns effectively. In this approach, the encoder compresses the sequential information into a compact hidden representation (hidden state), which reflects both structural and temporal relationships within the data. This hidden state is then transformed into an embedding vector that captures the node's dynamic characteristics over time. The decoder subsequently reconstructs the sequence, ensuring that both short-term and long-term dependencies are preserved in the representation. As a result, the LSTM Autoencoder produces numerical vector outputs for each node, combining their structural and temporal features. The flexibility of the LSTM Autoencoder allows it to model sequential data effectively, providing meaningful representations without solely depending on token embeddings.

In this study, we categorize embedding methods under three categories: NLP-based embedding methods, graph-based embedding methods, and sequence-to-sequence embedding methods. Word2Vec, an NLP-based embedding method, operates on linear sequences, predicting a word (or in this case, a page ID) based on its context within a sentence (or a session). Note that, here, the term prediction in the context of Word2Vec refers to the model's objective of estimating the likelihood of a target page ID given its surrounding pages within a session, following the Skip-Gram or CBOW architecture of Word2Vec. This approach captures semantic relationships by positioning similar items closer in the embedding space, making it particularly effective for textual and sequence-like data. In contrast, graph-based methods like Node2Vec, DeepWalk, and Struc2Vec extend embedding to graph-structured data. These methods simulate random walks on graphs to generate node sequences, which are then embedded using algorithms similar to Word2Vec. Sequence-to-sequence approaches, like LSTM autoencoders, further diverge by focusing on temporal dependencies in sequential data. These methods encode sequences into compact representations and reconstruct them, capturing complex temporal patterns and long-range dependencies. To this end, we argue that while NLP-based embedding methods excel in linear semantic representation, graph-based methods focus on structural relationships, and sequence-to-sequence approaches specialize in temporal dynamics.

Tokenization: In this study, we employed different tokenization schemes depending on the embedding method used, in order to capture user interactions effectively. For Word2Vec, we utilized page IDs as tokens [63]. This approach allows us to treat each page a user visits as a distinct "word" in the dataset, providing a meaningful representation of browsing behavior. The sequences of page IDs from user sessions were used as "sentences" to train the Word2Vec model, which helped us generate embeddings that capture relationships between pages based on user navigation.

For Node2Vec, Struc2Vec, and DeepWalk, the tokenization process involved treating web page visits as nodes in a graph. Each edge between nodes (pages) represents a transition in user behavior, such as moving from one page to another during a session. In this graph-based tokenization, nodes are the fundamental units, and sequences of edges (representing user navigation paths) form the basis for generating embeddings. These embeddings capture the structural patterns of user behavior across the entire network of interactions, providing insights into how users navigate the e-commerce platform.

For LSTM autoencoder, we adopted a sequence-to-sequence tokenization approach. Here, the tokens were derived from sequences of user interactions (page IDs) that preserved the temporal order of events. By tokenizing user sessions as ordered sequences, we leveraged the LSTM autoencoder's ability to model dependencies between successive actions, which was essential for capturing the temporal relationships and patterns in user behavior.

By using two tokenization schemes, one based on individual page IDs (for Word2Vec) and another on structured sequences or paths (for Node2Vec, Struc2Vec, DeepWalk, and LSTM Autoencoder), we ensured that our embedding methods were appropriately tailored to the nature of the data. This approach enabled us to generate embeddings that represent both isolated and sequential interactions, enhancing our ability to model user behavior comprehensively.

Weighting function module: This module incorporates a weighting algorithm responsible for executing the weighting function. In this algorithm, as shown in Algorithm 2, the Calculate EF sub-function computes the event frequency (EF) for the given page label within the session's page labels. The Calculate EF-ISF sub-function computes the EF-ISF for the given page label using both the session's page labels and the session collection. Here, the algorithm's if-else structure selects the appropriate weighting mechanism based on the weighting mechanism type parameter.

Module for session embedding vector calculation—No-weight approach: This module incorporates code segments that transform session-based clickstream data into numerical vectors using various embedding functions without employing a weighting mechanism.

The session matrix that can be obtained from a session-based clickstream data is shown in Equation (1),

$$S = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_M \end{bmatrix}, \tag{1}$$

ALGORITHM 2 | Weighting function.

Input: pageLabel, pageLabels, sessionCollection, weightingMechanismType.
Output: weight.

1. If weightingMechanismType == "No Weight":
   weight = 1.
2. Else if weightingMechanismType == "EF":
   weight = CalculateEF(pageLabel, pageLabels).
3. Else if weightingMechanismType == "EF-ISF":
   weight = CalculateEFISF(pageLabel, pageLabels, sessionCollection).
4. Else:
   Print "Error: Unknown weighting mechanism type" and exit.
5. Return weight.

where;

- S is a session matrix ($Nx1$) for sessions, obtained from a session-based clickstream data;
- Each row in the session matrix corresponds to a session vector S_i, where $\forall i \in \{1, 2, \ldots, M\}$;
- M is the total number of the sessions, obtained from the session-based clickstream data.

The embedding vectors of the pages in the session matrix are defined in Equation (2),

$$\mathrm{EV}_{Pj} = embed(p_j), \forall j \in \{1, 2, \ldots, N\} \tag{2}$$

where;

- EV_{Pj} embedding vector of the page ID of page j (p_j);
- j is the index for the page ID in a given session;
- N is the total number of unique page IDs that can appear in a clickstream data, and.
- $embed(p_j)$: Embedding function (such as word2vec, struc2vec, node2vec, deepwalk, LSTM Autoencoder).

The embedding vector of a session, which consists of the number of page IDs, is obtained as follows in Equation (3):

$$\mathrm{EV}_{Si} = \frac{\sum_{j=1}^{\mathrm{SessionLength}_i} \mathrm{EV}_{Pj}}{\mathrm{SessionLength}_i}, \forall i \in \{1, 2, \ldots, M\} \tag{3}$$

where;

- EV_{Si}: Embedding vector of session i;
- i is the index for the sessions in the log file (session-based clickstream data);
- EV_{Pj}: Embedding vector of the page ID of page j (p_j);
- j is the index for the page ID in the session i (S_i) and ranges from 1 to SessionLength of (S_i);
- $\mathrm{SessionLength}_i$: Total number of page IDs in the session i (S_i);
- M is the total number of the sessions, obtained from the session-based clickstream data.

Figure 2 depicts six sessions. In Figure 2a, the information, including session, page ID, and duration, is shown. In each session, the page IDs are organized in chronological order according to timestamp data, which indicates the duration of time spent on each website (See Figure 2b). Figure 2c depicts the page frequencies over all sessions. Figure 2 (d1, d2, d3, d4 and d5), depict embedding vectors for each page using Word2Vec, Struc2Vec, Node2Vec and DeepWalk and LSTM autoencoder embedding approaches. Figure 3 depicts the calculation of the embedding vector with no-weight for each session.

Module for weighted session-embedding vector calculation—EF-ISF Approach: This module incorporates code segments that transform session-based clickstream data into numerical vectors using various embedding functions, specifically utilizing the EF and EF-ISF weighting mechanisms. A complete description of these approaches is provided. Figures 4 and 5 are dedicated to the computation of weights used in weighted session embedding vector calculations.

Event Frequency equation is shown in Equation (4). Here, the main goal is to vectorize by assigning higher importance to the page ID that is visited most frequently in all sessions. Figure 4 depicts the process of calculating the Event Frequency-based weighted session embedding vector calculation.

$$\mathrm{EV}_{Si} = \frac{\left(\sum_{j=1}^{\mathrm{SessionLength}_i} \left(\frac{\mathrm{frequence}(p_j)}{\sum_{t=1}^{K} \mathrm{frequence}(p_t)}\right) * EV_{Pj}\right)}{\mathrm{SessionLength}_i}, \forall i \in \{1, 2, \ldots, M\} \tag{4}$$

where;

- EV_{Si}: Embedding vector of session i (S_i), $\forall i \in \{1, 2, \ldots, M\}$;
- EV_{Pj}: Embedding vector of the page ID j (p_j);
- i is the index for the sessions in the log file (session-based clickstream data);
- j is the index for the page IDs in the session i (S_i) and ranges from 1 to SessionLength of (S_i);
- M is the total number of the sessions, obtained from the session-based clickstream data;
- K is the total number of the unique page IDs that are available within the dataset;
- $frequence(p_j)$: Frequency function that gives the number of times page visit j (p_j) appears in the dataset;
- $\sum_{t=1}^{K} frequence(p_t)$: The total number of times page visits happened in all sessions, and
- $\mathrm{SessionLength}_i$: Total number of page visits in the Session i (S_i).

EF-ISF equation is shown in Equation (5), which is a weighting system obtained by multiplying the EF and ISF scores. Figure 5

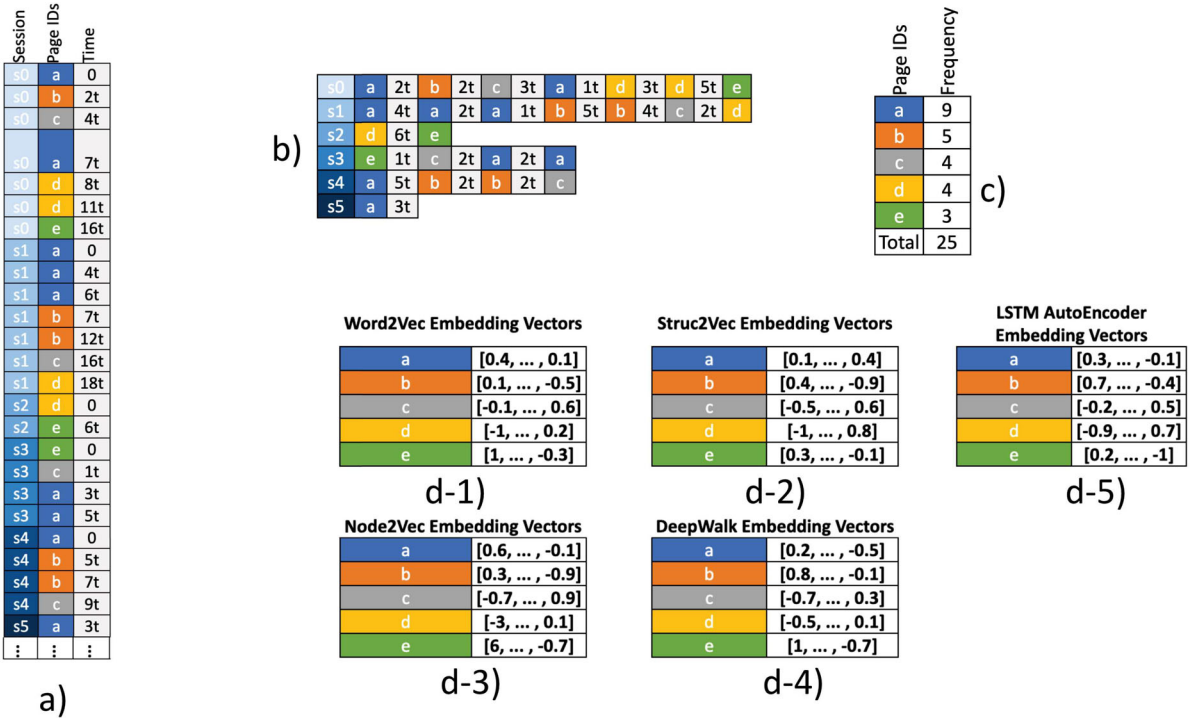

FIGURE 2 | Visualization of calculating weight-based vectorization.

	No-Weight Session Vectors **(No-Weight)**	**EV = Embedding Vector of Page Visit *j***
s0	$(EV_a + EV_b + EV_c + EV_a + EV_d + EV_d + EV_e) / 7$	
s1	$(EV_a + EV_a + EV_a + EV_b + EV_b + EV_c + EV_d) / 7$	
s2	$(EV_d + EV_e) / 2$	
s3	$(EV_e + EV_c + EV_a + EV_a) / 4$	
s4	$(EV_a + EV_b + EV_b + EV_c) / 4$	
s5	$(EV_a) / 1$	

FIGURE 3 | Visualization of calculating no-weight vectorization.

	Event Frequency Based Weighted Session Vectors **(EF)**	**EV = Embedding Vector of Page Visit *j***
s0	$(((9 / 25) * EV_a) + ((5 / 25) * EV_b) + ((4 / 25) * EV_c) + ((9 / 25) * EV_a) + ((4 / 25) * EV_d) + ((4 / 25) * EV_d) + ((3 / 25) * EV_e)) / 7$	
s1	$(((9 / 25) * EV_a) + ((9 / 25) * EV_a) + ((9 / 25) * EV_a) + ((5 / 25) * EV_b) + ((5 / 25) * EV_b) + ((4 / 25) * EV_c) + ((4 / 25) * EV_d)) / 7$	
s2	$(((4 / 25) * EV_d) + ((3 / 25) * EV_e)) / 2$	
s3	$(((3 / 25) * EV_e) + ((4 / 25) * EV_c) + ((9 / 25) * EV_a) + ((9 / 25) * EV_a)) / 4$	
s4	$(((9 / 25) * EV_a) + ((5 / 25) * EV_b) + ((5 / 25) * EV_b) + ((4 / 25) * EV_c)) / 4$	
s5	$(9 / 25 * EV_a) / 1$	

FIGURE 4 | Visualization of calculating EF vectorization.

	Event Frequency - Inverse Session Frequency Based Weighted Session Vectors **(EF-ISF)**	**EV = Embedding Vector of Page Visit *j***
s0	$(((9 / 25) * \ln(6 / (5 + 1)) * EV_a) + ((5 / 25) * \ln(6 / (3 + 1)) * EV_b) + ((4 / 25) * \ln(6 / (4 + 1)) * EV_c) + ((9 / 25) * \ln(6 / (5 + 1)) * EV_a) + ((4 / 25) * \ln(6 / (3 + 1)) * EV_d) + ((4 / 25) * \ln(6 / (3 + 1)) * EV_d) + ((3 / 25) * \ln(6 / (2 + 1)) * EV_e)) / 7$	
s1	$(((9 / 25) * \ln(6 / (5 + 1)) * EV_a) + ((9 / 25) * \ln(6 / (5 + 1)) * EV_a) + ((9 / 25) * \ln(6 / (5 + 1)) * EV_a) + ((5 / 25) * \ln(6 / (3 + 1)) * EV_b) + ((5 / 25) * \ln(6 / (3 + 1)) * EV_b) + ((4 / 25) * \ln(6 / (4 + 1)) * EV_c) + ((4 / 25) * \ln(6 / (3 + 1)) * EV_d)) / 7$	
s2	$(((4 / 25) * \ln(6 / (3 + 1)) * EV_d) + ((3 / 25) * \ln(6 / (2 + 1)) * EV_e)) / 2$	
s3	$(((3 / 25) * \ln(6 / (2 + 1)) * EV_e) + ((4 / 25) * \ln(6 / (4 + 1)) * EV_c) + ((9 / 25) * \ln(6 / (5 + 1)) * EV_a) + ((9 / 25) * \ln(6 / (5 + 1)) * EV_a)) / 4$	
s4	$(((9 / 25) * \ln(6 / (5 + 1)) * EV_a) + ((5 / 25) * \ln(6 / (3 + 1)) * EV_b) + ((5 / 25) * \ln(6 / (3 + 1)) * EV_b) + ((4 / 25) * \ln(6 / (4 + 1)) * EV_c)) / 4$	
s5	$((9 / 25) * \ln(6 / (5 + 1)) * EV_a) / 1$	

FIGURE 5 | Visualization of calculating EF-ISF vectorization.

provides a full explanation of the EF-ISF computation.

$$EV_{Si} = \frac{\left(\sum_{j=1}^{\text{SessionLength}_i}\left(\left(\frac{frequence(p_j)}{\sum_{t=1}^{K} frequence(p_t)}\right) * \left(ln\left(\frac{M}{\sum_{k=1}^{M} f_j(k)+1}\right)\right) * EV_{Pj}\right)\right)}{\text{SessionLength}_i}, \quad \forall i \in \{1, 2, \ldots, M\} \tag{5}$$

where;

- EV_{Si}: Embedding vector of Session i (S_i), $\forall i \in \{1, 2, \ldots, M\}$;
- EV_{Pj}: Embedding vector of the page ID j (p_j);
- i is the index for the sessions in the log file (session-based clickstream data);
- j is the index for the page visits in the Session i (S_i) and ranges from 1 to SessionLength of (S_i);
- M is the total number of the sessions, obtained from the session-based clickstream data;
- $f_j(k)$: Binary indicator function that determines whether a specific page visit j is part of given session k;
- $\sum_{k=1}^{M} f_j(k)$: The total number of sessions that contain the page visit j;
- K is the total number of the unique page visits (page IDs) that are available within the dataset;
- $frequence(p_j)$: Frequency function that gives the number of times page visit j (p_j) appears in the dataset;
- $\sum_{t=1}^{K} frequence(p_t)$: The total number of times page visits (page IDs) happened in all sessions, and
- SessionLength_i: Total number of page visits (page IDs) in the session i (S_i).

The EF-ISF weighting approach extends the concept of TF-IDF specifically for session-based clickstream data in the e-commerce domain by incorporating session-specific dynamics and temporal patterns. When calculating the occurrence frequency, the TF-IDF approach takes into account the occurrence frequency of terms within individual documents, while EF-ISF takes into account the occurrence frequency of events across whole datasets. Hence, EF-ISF accounts for the frequency of events and their rarity across whole sessions, making it especially suitable for sequential datasets such as clickstream data. In this context, we emphasize that while our approach leverages the foundational concept of TF-IDF, it introduces a novel weighting mechanism tailored to session-based clickstream data. This mechanism weights events based on their frequency within whole available sessions and their rarity across the entire dataset, with a particular focus on graph and sequence-to-sequence embedding spaces.

Weighted Average Based Vectorization Module: This module is responsible for creating the weighted average vectorization. In this pseudo-code given as Algorithm 3, the embedding function chooses the appropriate embedding method to convert a page label into a numerical vector based on the embedding function type. Each case within the switch statement corresponds to a different embedding algorithm, calling the relevant function to generate the numerical vector for the given page Label.

ALGORITHM 3 | Weighted average vectorization of browsing behavior.

Input: pageLabels, sessionCollection, embeddingFunctionType, weightingMechanismType.
Output: weightedAverageVector.

1. Initialize sumVector as a vector of size N with all values as 0.
2. Set totalWeight = 0.
3. For each pageLabel in pageLabels:
   (a) numericalVector = EmbeddingFunction(pageLabel, embeddingFunctionType)
   (b) weight = WeightingFunction(pageLabel, pageLabels, sessionCollection, weightingMechanismType)
   (c) For each i from 0 to $N-1$:
      (i) sumVector[i] += weight * numericalVector[i].
   (d) totalWeight += weight.
4. If totalWeight âL'ă 0:
   (a) For each i from 0 to $N-1$:
      (i) weightedAverageVector[i] = sumVector[i] / totalWeight.
   (b) Return weightedAverageVector.
5. Else:
   (a) Print "Error: Total weight is zero" and exit.

Clustering Module: This module employs three different clustering algorithms. Clustering algorithms are essential in many data analysis tasks, dividing data points into groups or clusters based on their commonalities. The first is the K-Means algorithm, which is an assignment-based clustering method. The second algorithm is DBSCAN, which is a density-based clustering method. The last algorithm is the Gaussian Mixture Model, which is designed to combine multidimensional Gaussian probability distributions.

Known as an assignment-based clustering algorithm, the K-Means algorithm assigns data points to clusters iteratively until a convergence criterion is achieved. The algorithm begins with the selection of k centroids randomly, where k represents the number of clusters to be constructed. Each data point is iteratively assigned to the nearest centroid and the centroid location is updated by taking the mean of the data points assigned to that centroid. The process is repeated until the centroids have stopped moving or a maximum number of iterations has been reached.

On the other hand, the DBSCAN algorithm utilizes a density-based clustering approach. Data points that are close to each other in terms of distance and density are grouped, while points in low-density regions are divided. The algorithm begins by selecting a data point at random and checking whether it has at least a minimum number of neighbors within a specific radius. In the presence of sufficient neighbors, the algorithm creates a cluster around the initial data point. Whenever new points meet

the density and distance requirements, they are added to the cluster. If no more points meet the requirements, the algorithm selects a new unvisited data point and repeats the process until all data points have been visited.

GMM is a probabilistic clustering and density estimation model. Data points in GMM are assumed to be generated by the mixture of different Gaussian distributions and the parameters of these distributions are estimated to fit the data.

All three algorithms have strengths and weaknesses and can be used depending on the problem at hand. The K-Means algorithm, for example, is simple and fast, but it assumes that clusters are spherical and of equal size. The DBSCAN algorithm, on the other hand, can handle clusters of varying shapes and sizes, but it requires the parameters for the minimum density and radius to be set appropriately. GMM provides flexibility and probabilistic modeling capabilities. However it is sensitive to initialization, computationally demanding, requires a priori knowledge of cluster counts, and assumes Gaussian data distributions.

Clustering result evaluator module: This module is responsible for evaluating clustering algorithms. Assessing the efficacy of clustering algorithms is crucial in determining their ability to organize data into meaningful clusters. In this study, we utilize two different clustering quality evaluation metrics: The silhouette coefficient and The DBI.

The silhouette coefficient, also known as the *Silhouette Coefficient Score*, is a commonly used metric for evaluating clustering efficiency. The silhouette coefficient quantifies the degree of separation between clusters. It considers both the internal consistency within groups and the distinctiveness between groups. The silhouette coefficient score is a numerical measure that falls within the range of −1 to 1. A score closer to 1 implies that the data points are effectively clustered, with clearly defined and well-separated clusters. On the other hand, a score closer to −1 indicates that the data points might have been incorrectly assigned to clusters, indicating the presence of overlapping or poorly defined borders. A score of 0 suggests the presence of clusters that overlap or contain data points that are close to the decision border.

Equation (6) denotes the mathematical expression employed to compute the silhouette coefficient score to assess clustering techniques.

$$\text{Silhouette coefficient score} = \frac{b - a}{\max(a, b)} \tag{6}$$

where, a is the average intra-cluster distance (the average distance between points) distance, and b is the average inter-cluster distance (the average distance between all clusters) distance.

The DBI is another important metric for measuring clustering efficacy. It provides insight into the cohesion of clusters. DBI assigns a numerical value to the average similarity ratio between each cluster and its most analogous cluster. This metric is founded on the concept of intra- and inter-cluster distances. A reduced DBI indicates superior clustering performance. Utilizing the DBI in our research will enhance our understanding of the clustering structure and the efficacy of the methods. Equation (7) denotes the mathematical expression employed to obtain the DBI score to assess the clustering techniques.

$$\text{Davies} - \text{Bouldin Index} = \frac{1}{n}\sum_{i=1}^{n}\max_{j\neq i}\left(\frac{s_i + s_j}{d_{ij}}\right) \tag{7}$$

where;

- n: Number of clusters.
- s_i: A measure of the internal similarity of the i-th cluster. It typically represents the average distance of the points in the i-th cluster from the cluster centroid (mean or median). A smaller value of s_i indicates that the points within the cluster are closer to each other, indicating higher cohesion.
- d_{ij}: Distance between the centroids of the i-th and j-th clusters. It reflects the separation between the clusters; a larger value indicates that the clusters are well-separated.

When the DBI looks at two groups, it looks at the ratio of the sum of their internal similarities to the distance between them. We take the average of the index over all the clusters and look at the highest ratio number for each cluster. When the DBI number is low, clustering works better because the clusters are more distinct from each other and also more compact inside. To sum up, the DBI gives information about the general quality of clustering by balancing how close clusters are to each other and how far apart they are from each other.

5 | Implementation and Evaluation of the Proposed Business Process

This section describes of the datasets used in this study and presents the implementation and evaluation of the business process recommended in Figure 1. The primary aim of this section is to assess the practical application of the EF-ISF-based weighting mechanism across different datasets. By using established methodologies, we implement the process across multiple datasets to evaluate its effectiveness in capturing user behavior patterns. We focus on various stages of data preprocessing, embedding techniques, and clustering methods, ensuring that the business process is applied consistently to each dataset. The results of this implementation are analyzed to highlight the strengths and limitations of the proposed approach, providing insights into its adaptability and performance in diverse contexts.

5.1 | Datasets

The datasets used in this study were carefully selected to ensure a comprehensive evaluation of the proposed approach across different e-commerce platforms. Each dataset presents distinct characteristics, offering a robust testbed for analyzing the effectiveness of the EF-ISF weighting mechanism in representing and clustering user behavior. In the following subsections, the specifics of the two datasets are detailed, including their size, structure, and key attributes. We argue that these datasets, sourced from publicly available repositories, provide valuable insights into user engagement and navigational patterns, enabling an analysis of the proposed methodology.

5.1.1 | Dataset 1

The dataset used in this study is available on the Kaggle site [64]. The dataset used in this study has a file size of 6.5 GB and consists of 20 columns and 9,313,563 rows.

The dataset utilized encompasses pivotal attributes delineating user engagement and interaction dynamics. The count of distinct customers stands at 654.897, indicative of the unique individuals participating in the observed activities. A total of 197.518 sessions were recorded over a 96-unit period, providing insights into the overall frequency of user engagements. In terms of visiting page counts per session, the dataset reflects variability, ranging from a minimum of 1 to a maximum of 56, elucidating diverse user exploration behaviors. Furthermore, the temporal dimension is characterized by the maximum and minimum time spent on a page, spanning from 1 to 1402 s. Notably, the dataset also features information on the number of large-scale URLs, revealing the presence of 233.159 such URLs within the dataset. These comprehensive metrics collectively contribute to a nuanced understanding of user behavior, facilitating profound analyses and interpretations of the dataset's intrinsic characteristics. This detailed analysis is shown in Table 1.

To ascertain the distribution of events within individual sessions, a thorough analysis of the session-based dataset that was provided was conducted for this study. After a thorough analysis of the dataset, the number of occurrences each session was determined. The frequency distribution of sessions containing 1–6, 7, 8, 8–10, and more than 10 occurrences or more was then graphically represented. Figure 6 provides a visual representation of the observed patterns. In addition to providing insights into the structural makeup of the dataset, it serves as an important point of reference for understanding the differences in session lengths.

TABLE 1 | Dataset features.

	Dataset 1	Dataset 2
Number of distinct customers	654.897	96.200
Number of total sessions	197.518	296.100
Number of large-scale URLs	233.159	46
Unique event count	9	46
Average number of events per session	47, 15	8, 61
Maximum number of events per session	15.221	11
Minimum number of events per session	1	1
Total number of events	9,313,563	43,390,766
Time period for the dataset	96	14
Max. and min. visiting page count per session	Min = 1, Max = 56	Min = 1, Max = 11
Max. and min. time spent on a page	Min = 1 s, Max = 1402 s	Min = 1 s, Max = 1349 s
Average time spent in session	7.680, 87 s	4.976, 34 s

To increase the reliability and accuracy of the clustering results, sequences with a minimum of eight occurrences were selected for embedding. This standard was created to reduce the noise that might come from shorter sequences, which might not catch the important patterns well enough. We wanted to make sure that the model could learn from a more stable dataset by focusing on these longer patterns. This would improve its performance and ability to be used in other situations. A better knowledge of how the sessions work is made possible by this approach, which also makes the methods used in this study more stable.

The dataset under consideration encompasses sessions, collectively spanning a duration of 96 days. Each session records a minimum duration of 1 s, with the maximum session duration reaching 4406.171 s. The dataset's temporal landscape is characterized by an average session duration of 7680.87 s. This temporal diversity, ranging from brief interactions to more extensive engagements, poses a rich ground for exploring clustering methodologies in the context of embedding techniques. The varied session durations present an opportunity to delve into the nuanced patterns and structures inherent in the dataset, offering insights into user behaviors and interactions over the 96-day period.

5.1.2 | Dataset 2

In this study, we used data, which is available on Kaggle site [65], from a mobile app for a chain of coffee shops. The dataset used has its own set of characteristics that show how users interacted and engaged with the data over 14 days. This dataset has 96,200 unique customers, which are the unique people whose actions were tracked. During the study, 296.100 sessions were recorded, which gives us a solid base for looking at patterns of user involvement.

When it comes to how users explore, each session has anywhere from 1 to 11 page views, which shows how different users' interactions are. Users spent anywhere from 1 s to a maximum of 1.349 s on each page, which shows how different the session lengths are. This dataset also records a lot of information about user activities. It keeps track of 43,390.766 events, with an average of 8.61 events per session and a maximum of 11 events in a single session. There are 46 large-scale URLs in the dataset, which gives us more information about the types of exchanges we observe. This dataset's average session length is 4.976.34 s, which gives a clear picture of how users behave over time. Table 1 shows how these measures can be used to fully explore the dataset. This makes it easier to get a deep understanding of how users interact with the dataset and how it changes over time. In order to make the clustering results more reliable and accurate, embedding was applied by selecting events that were repeated at least eight times. This standard was created to reduce noise due to the risk of shorter sequences not capturing important patterns sufficiently. By focusing on longer patterns, it was desired to ensure that the model learned from a more stable dataset.

The session-based dataset was looked at in great depth to find out how the events were spread out within each session. The number of times something happened in a session was found through this study. The sessions were put into groups based on

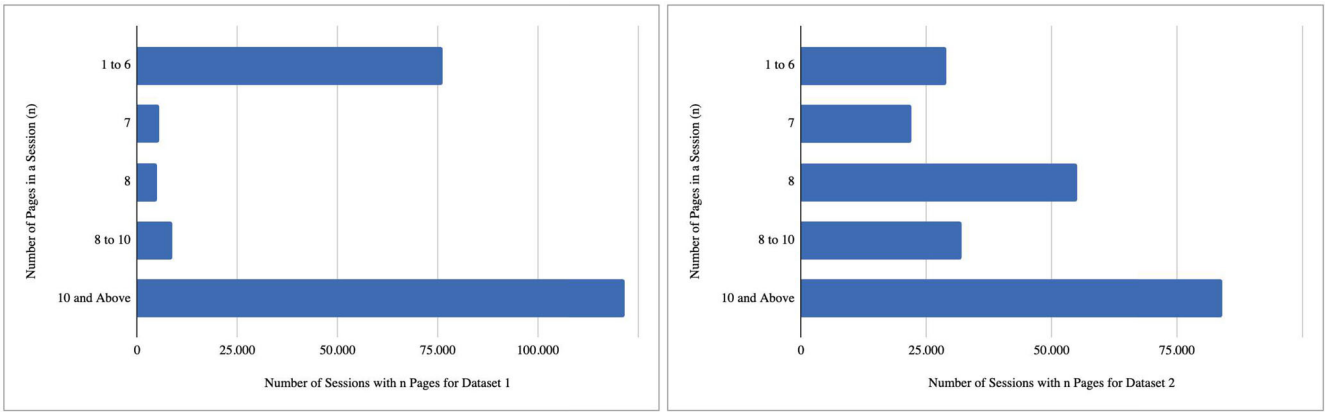

FIGURE 6 | Number of sessions with event counts for datasets.

how many events they had. There were groups for sessions with 1–6 events, 7 events, 8 events, 8–10 events, and more than 10 events. After that, these groups were displayed to show how the data was distributed. This is shown in Figure 6, which also shows how the lengths of sessions change over time. This analysis not only sheds light on the internal structure of the dataset, but it also helps us understand why session lengths and event occurrences are different.

5.2 | Implementation

The proposed business process was implemented with the version 3.10.4 of the Python programming language using the Gensim library [66] version 4.2.0 for the Word2Vec embedding method. As part of the study, Apache Spark 3.2.3 was also utilized. The library in [67, 68] was utilized to construct graphs with the Struc2Vec, DeepWalk, Node2Vec and LSTM Autoencoder algorithms.

Despite the availability of specialized libraries for the Node2Vec algorithm, we chose to develop the necessary steps of the algorithm ourselves. The library [3] was used to train the generated graph and obtain an edge model. By using the node2vec library to train the data, the resulting model can produce the vectors needed for clustering.

A dedicated function of the DeepWalk algorithm develops random walk sequences based on the graph structure obtained through NetworkX. It facilitates the generation of node sequences encountered during random walks initiated from a specified node. Random walks are then systematically computed for each node in the graph.

The subsequent random walk sequences were trained using the Word2Vec embedding algorithm. The training process produced several useful outcomes, including the identification of nodes that are located close to one another within the graph. The Word2Vec algorithm was then used to obtain vector representations.

While using the Struc2Vec; a basic undirected graph is created using the NetworkX library, representing nodes and edges. The Struc2Vec algorithm is encapsulated in the Struc2Vec class, allowing for customization of parameters such as dimensions, walk length, number of walks, window size, and workers. The algorithm generates random walks within the graph, and the Word2Vec model from the Gensim library is employed to train Struc2Vec, learning embeddings for each node.

Before using the LSTM autoencoder embedding method, the NetworkX tool was used to create a graph. Subsequently, an adjacency matrix was constructed from the characteristics of the nodes and edges to appropriately form the graph. This matrix was input into the LSTM autoencoder model. The adjacency matrix was provided to the input layer by configuring the LSTM autoencoder using the Keras deep learning framework. The embedding vectors from the encoder part were used to obtain a representation of the nodes after the model had been trained.

The weight of each page of a single sequence was determined to create a single vector for each session. Then, the weighted session embedding vectors were calculated by taking a weighted average of the page embedding vectors.

5.3 | Evaluation

In the clustering phase, the k value for the K-Means and Gaussian Mixture Model algorithms was determined using the Elbow method and Silhouette method. For K-means clustering, the Elbow method is often used to determine the number of groups, or clusters, k, in a dataset. An "elbow" point is found when the within-cluster sum of squares (WCSS) is plotted against k. WCSS, also known as intra-cluster variance, is a measure used to assess the homogeneity and compactness of clusters. Data points within each cluster are quantified based on their dispersion. An elbow point indicates that clustering has diminishing returns, and it is also the point at which the rate of WCSS decreases rapidly. The Silhouette Method was utilized to determine the k values for the K-Means, DBSCAN, and Gaussian Mixture Model (GMM) clustering algorithms for each embedding method. In the Silhouette method, the Silhouette score is calculated for each k number to determine the quality of the clustering. A peak Silhouette score indicates that the data is divided into well-defined clusters, indicating that the data is well-defined. The optimum k number for our dataset is found as 4, as depicted in Figure 7. Together, the Elbow and Silhouette Methods provide a complete picture of the

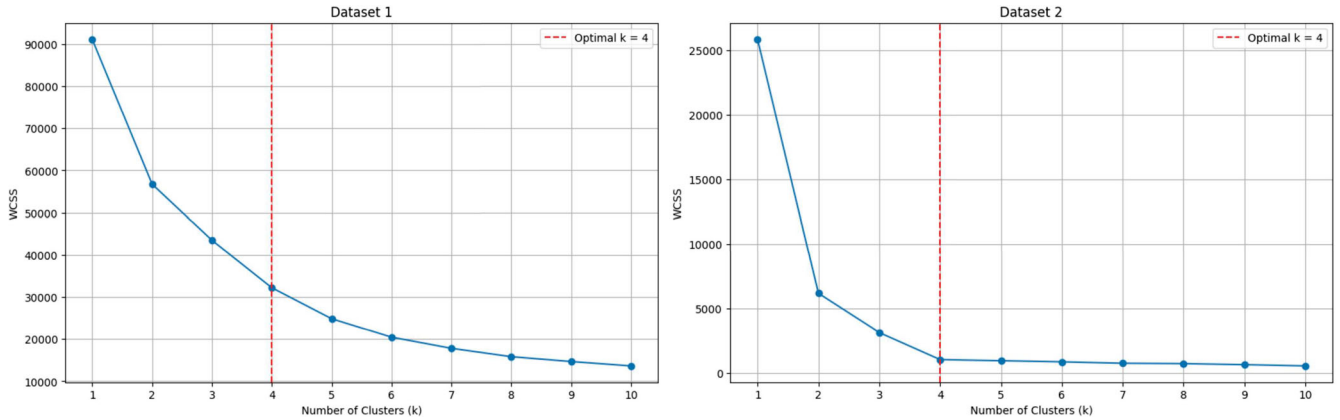

FIGURE 7 | Optimum k value with elbow method.

right number of clusters, ensuring that the clustering results are accurate and well-informed. The same k values were obtained using both methods.

The number of clusters for the DBSCAN algorithm was determined as follows:

- Density-dependent points were input to a recursive function. Points refer to specific locations within a space in mathematics or geometry. In most cases, points are represented by a set of coordinates. Upon determining their locations, all points were assigned the same cluster as the core point.
- For two points to be density-connected, there must be at least one density-reachable point between them. According to density-based clustering algorithms, such as DBSCAN, density-reachability refers to the property that a point in a dataset can be reached from another point if there is a chain of nearby points within a specified distance threshold. A density-reachable point can be reached by a chain of neighboring points within a given eps distance. In clustering algorithms, epsilon distances (eps) are used to determine the proximity of points to one another. If two or more points in a dataset have at least one density-reachable point in common within an eps distance, they are density-connected. As an example, assume there are five points in a dataset, namely a, b, c, d, and e. It can be inferred that if b is a neighbor of a, and if a is a neighbor of c, c is a neighbor of d, and d is a neighbor of e, then b is density-reachable from e because there is a chain of neighboring points connecting them. If a and b share common density-reachable points, such as e, it can be inferred that a and b are density-connected.
- Points with density connections form clusters, while points without density connections may be regarded as noise or outliers, and the remaining unexplored points within the dataset are located and visited.
- As a result, all points are marked and clustered.

The effect of weighted embedding strategies for the session data on the quality of clustering algorithms is shown in Figures 8 and 9 for dataset 1 and dataset 2, respectively. To this end, we used three different weighting mechanisms, namely no-weight, EF and EF-ISF. The Silhouette coefficient metric and DBI were used to measure the results. When looking at the graphs, it's important to keep in mind that the DBI has a score that is opposite to the Silhouette index: it shows better clustering at numbers closer to 0. Figures 8 and 9 depicting the clustering results, show that the EF-ISF balancing method performs better compared to other methods.

The Silhouette and DBI scores of the LSTM Autoencoder embedding method using different weighting techniques in a sequence-to-sequence setting are shown in Figures 10, 11. Scores improve as the weighting method changes from No-Weight to EF-ISF. In general, the K-Means clustering method provides the best results for clustering. The LSTM Autoencoder method yields the best outcomes for both Dataset 1 and Dataset 2, indicating consistent findings across both files when compared.

When comparing the K-Means algorithm against the DBSCAN and GMM in the Word2Vec embedding method, we observe that K-means has the highest performance. Also, changing the weighting from no-weight to EF-ISF is seen to improve the general performance of clustering when using the Word2Vec embedding approach. When the K-Means method is used, the clustering results (for EF-ISF weighting mechanism) show that the data is more meaningfully and uniformly grouped. On the other hand, when DBSCAN was used for analysis, the noise and density-based clustering became clearer. In tests using the GMM algorithm, EF-ISF was observed to yield better data distribution and more accurate results. Similarly, the K-Means algorithm performs better in the Node2Vec method, showing the usefulness of the EF-ISF weighting mechanism. The K-Means algorithm worked best in the Struc2Vec method, and the clustering performance for EF-ISF improved compared to No-Weight.

The no-weight and EF gave similar results in the DeepWalk embedding method, but the EF-ISF weighting stood out. In this case, it shows how well EF-ISF works with this embedding method compared to other weighting mechanisms.

This study shows that the EF-ISF weighting method always performed the best at grouping the session data. The results show that EF-ISF based weighting mechanism works well with unsupervised clustering algorithms and embedding methods for the datasets. In response to first research question, based on the

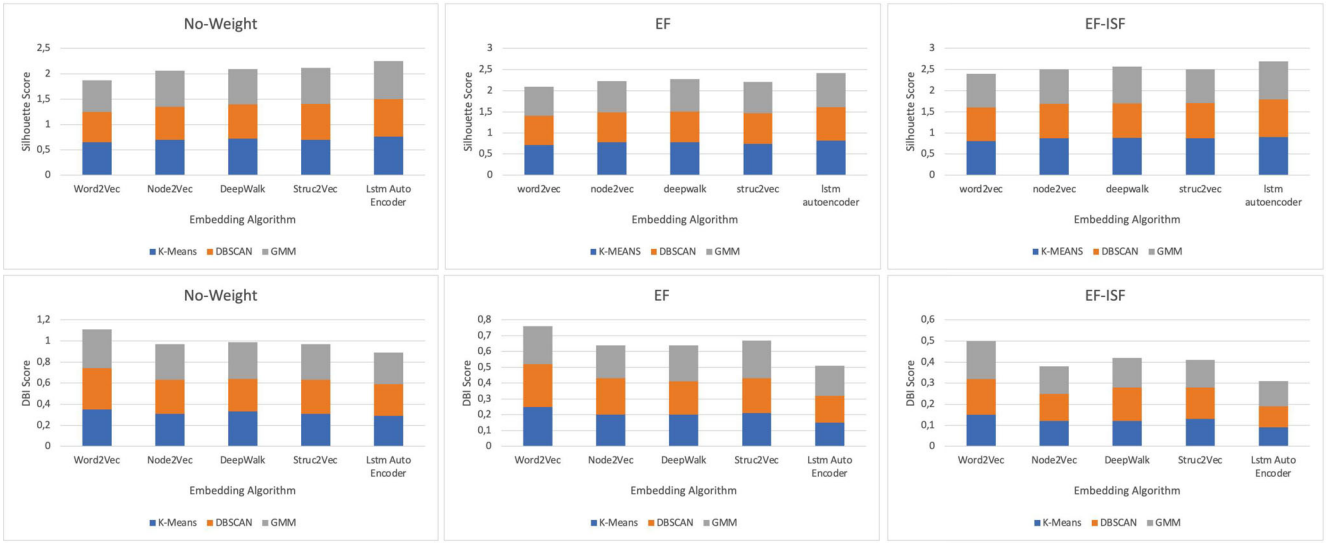

FIGURE 8 | Clustering quality results (Silhouette coefficient and Davies–Bouldin index) across different embedding methods with different weighting mechanisms for dataset 1.

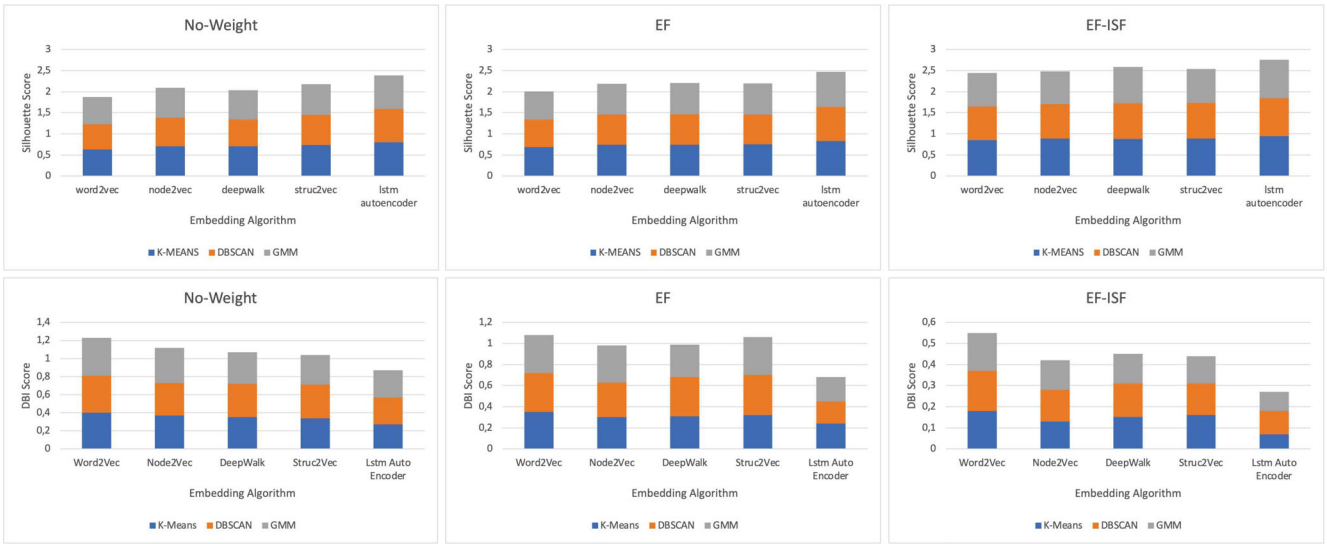

FIGURE 9 | Clustering quality results (Silhouette coefficient and Davies–Bouldin index) across different embedding methods with different weighting mechanisms for dataset 2.

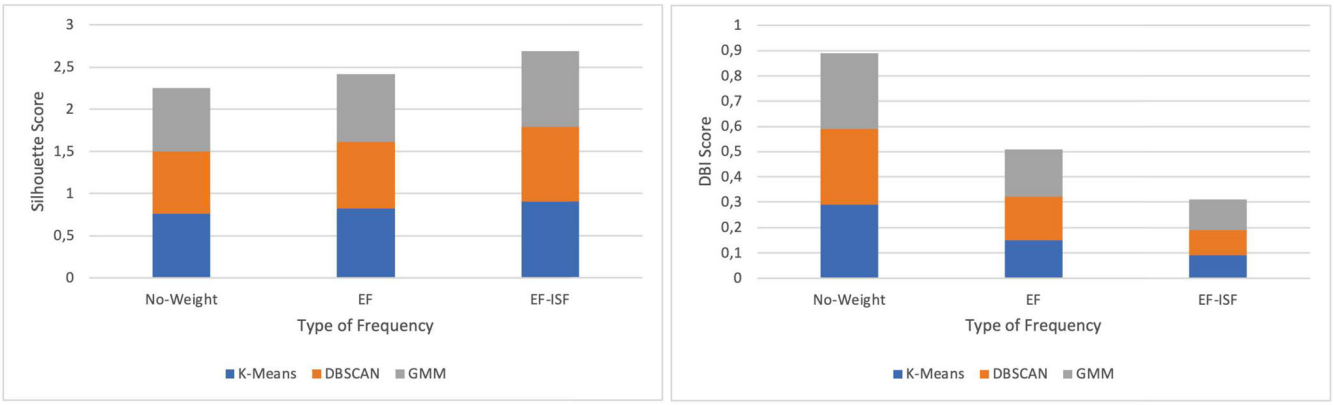

FIGURE 10 | Clustering quality results (Silhouette coefficient and Davies–Bouldin index) across different embedding methods with different weighting mechanisms for dataset 1.

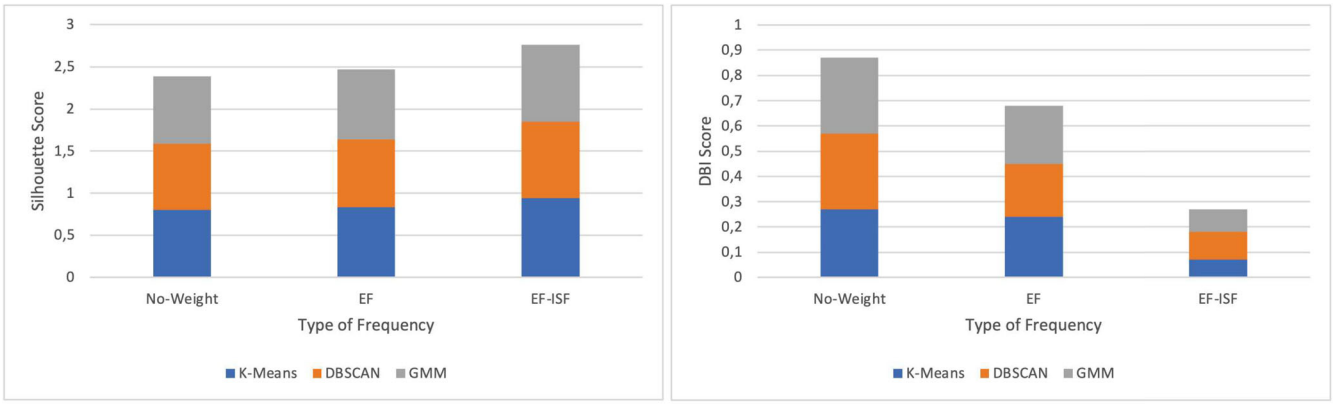

FIGURE 11 | Clustering quality results (Silhouette coefficient and Davies–Bouldin index) for LSTM autoencoder embedding method with different weighting mechanisms for dataset 2.

results, we argue that he integration of the EF-ISF weighting mechanism with traditional embedding methodologies significantly enhances the accuracy and depth of user navigational behavior analysis by ensuring that embeddings capture the nuanced interactions users have on e-commerce platforms. In response to second research question, we argue that, incorporating the EF-ISF weighting mechanism enhances the quality and interpretability of embeddings generated from user clickstream data by adjusting the significance of events within user sessions, thereby leading to more informative and representative embeddings.

The use of K-Means clustering to Word2Vec embeddings resulted in clusters that had a higher degree of cohesion and semantic significance in comparison to the baseline without any weights. The clustering process was further improved by using EF, with the most notable enhancement observed with the utilization of EF-ISF weighting. This implies that the algorithm gains advantages from the careful evaluation of the significance of terms in the document corpus.

Node2Vec exhibited improved clustering capabilities when weightings were incorporated, similar to Word2Vec. The Node2Vec embeddings produced clusters that were more cohesive and delineated, especially when using EF-ISF weighting. This suggests that it is essential to capture the complex connections between nodes in a network structure, and account the significance of each term enhances the algorithm's effectiveness.

The DeepWalk embeddings demonstrated a considerable improvement in cluster quality as the weightings were examined. The algorithm demonstrated its capacity to effectively capture the proximity of nodes in graph structures, which was further enhanced by including EF-ISF weighting. The semantic coherence and separation of the clusters derived from DeepWalk embeddings under EF-ISF weighting were found to be higher.

The application of various weightings resulted in a significant enhancement in the clustering quality of Struc2Vec embeddings. The utilization of EF-ISF weighting yielded clusters that exhibited both distinctiveness and semantic richness. The results indicate that the utilization of a weighted embedding strategy in Struc2Vec yields notable advantages, highlighting the significance of term importance in representations based on structural graphs.

The LSTM Autoencoder outperforms Word2Vec, Node2Vec, DeepWalk, and Struc2Vec in representing the data compared to the other four embedding techniques. The grouping results indicate that weighting strategies significantly impact outcomes. The procedure improved significantly from a no-weight approach as more sophisticated weighting techniques, such as EF and EF-ISF, were implemented. The findings, particularly with the application of EF-ISF scaling, indicated that the LSTM Autoencoder outperforms alternative methodologies. K-Means exhibits the highest stability and success rate among clustering methods.

The combination of the LSTM Autoencoder and EF-ISF weighting yielded optimal outcomes for data representation and clustering. This performs effectively as EF-ISF accentuates the significance of words or features. This allows sequence-to-sequence embedding methods to make representations that are more meaningful and useful. EF-ISF is better at grouping sequences together because it better shows the context and links of the data. Sequence-to-sequence embedding methods are also successful at capturing complex relationships and serial dependencies. This helps make clusters richer and more meaningful when combined with EF-ISF.

In response to third research question, based on the results, we argue that the integration of the EF-ISF weighting with various embedding techniques, including Word2Vec and LSTM Autoencoder, has been shown to improve clustering performance significantly, as evidenced by enhanced clustering cohesion and distinctiveness in user navigational patterns.

The long-standing effectiveness of EF-ISF weighting highlights its ability to accurately capture the subtle semantic details included in the embedding space. We argue that this research offers significant insights for scholars and professionals who are interested in identifying the most effective configurations for embedding algorithms in clustering tasks.

The results clearly show that, for data representation, the integration of the EF-ISF-based weighting mechanism and the embedding functions is better at understanding how people navigate

TABLE 2 | Performance improvement (%) of K-Means clustering algorithm and graph-based (node2vec) and seq-to-seq (LSTM autoencoder) embedding methods with integrated weighting mechanisms compared to no-weight baseline.

K-Means (Silhouette)	Node2vec		LSTM autoencoder	
	Dataset 1 (%)	Dataset 2 (%)	Dataset 1 (%)	Dataset 2 (%)
EF	11.42	12.12	14.47	7.5
EF-ISF	24.28	22.72	19.73	17.5

e-commerce sites than older embedded methods. Dataset 1 and Dataset 2 show how users interacted with the e-commerce system, since they exhibit similar patterns of behavior. In response to fourth research question, based on the results, we argue that when applied to the two distinct datasets, the EF-ISF weighting mechanism demonstrated consistent performance in detecting distributions and clusters across dynamic and sequential event characteristics, affirming its robustness across different data types. The EF-ISF mechanism can consistently record users' navigation habits on various e-commerce platforms, and both datasets usually show similar user behaviors. Moreover, the EF-ISF-based method showed better performance compared to standard embedded methods, showing higher accuracy and recall rates. The outcomes demonstrate that the EF-ISF-based weighting method can enhance e-commerce platforms' recommendation systems by providing a deeper understanding of how users interact with them. This can also have a positive impact on conversion rates by raising user satisfaction. According to the results of both datasets, this study shows that the EF-ISF-based weighting method is a powerful way to improve recommendation systems in the e-commerce domain.

Based on the results, graph-based embedding methods are observed to be more effective at representing data, especially when it comes to capturing the intricate details of how users browse and navigate websites. LSTM Autoencoder has a clear advantage over both graph-based and NLP-based embedding methods. In response to fifth research question, based on the results, we argue that the LSTM Autoencoder outperformed both graph-based and NLP-based embedding methods in data representation, indicating that its sequential nature captures user navigational behavior more effectively.

Table 2 shows the performance evaluation of clustering algorithms and weighting methods compared to the no-weight method. The performance improvement compared to the no-weight frequency method is shown in percentages. The K-Means clustering method was chosen for testing since it provides the highest clustering performance. The Silhouette Score was used to measure the strength of the clustering. Most of the embedding algorithms outperformed the Word2Vec embedding. Other graph-based algorithms, such as DeepWalk and Struc2Vec, also displayed higher performances. A sequence-to-sequence approach known as LSTM Autoencoder was employed in the test to demonstrate its efficacy. Table 2 shows that the EF-ISF weighting method outperforms EF. The LSTM Autoencoder algorithm outperforms the other embedding methods. The performance gains were observed to be consistent, hence the embedding algorithms and weighting methods can effectively be used for interpreting users' digital footprints and habits. It was also observed that unsupervised ML approaches significantly enhance the performance.

Based on the results from Table 2, for e-commerce website log file data (such as Dataset-1), where page content is dynamic and users frequently refresh pages (leading to event repetitions), our experimental study shows that the success of our proposed weighting mechanism based on event frequency. This mechanism proved effective in detecting distributions and clusters within the data. In datasets, (such as Dataset-2) which are composed of sequential events with temporal dependencies, such as web session log files, our sequence-to-sequence embedding approach also performed well in identifying patterns and groups. The effectiveness of our proposed weighting approach further enhanced the identification of these distributions. Additionally, for websites where users perform distinct actions, leading to sparse distributions of events across sessions (i.e., rare event types in sessions), our EF-ISF weighting mechanism successfully identified the underlying distributions and clusters.

6 | Conclusion and Future Work

In this study, we introduced a novel approach for analyzing user navigational behavior on e-commerce platforms through an EF-ISF based weighting mechanism integrated into data embedding methodologies. By applying this mechanism to a clickstream dataset from an e-commerce website, we demonstrated how EF-ISF weighting enhances the representation and clustering of user behaviors. This approach leverages the strengths of unsupervised machine learning algorithms and embedding techniques, such as Word2Vec, Node2Vec, DeepWalk, Struc2Vec and, LSTM autoencoder, offering a more granular and accurate analysis of user navigational patterns.

The experimental results affirm the efficacy of incorporating EF-ISF weighting in improving the clustering outcomes, as evidenced by the improved Silhouette coefficient scores and DBI scores. This improvement suggests that EF-ISF weighting not only refines the embeddings but also contributes to a more discerning and meaningful segmentation of users' browsing behaviors. In the case where page contents change dynamically and users refresh these pages frequently, EF-ISF weighting was observed to be successful in distinguishing clusters and groups of data. Similar conclusions were observed for web pages containing ordered time-correlated events. Additionally, for websites characterized by distinct user actions leading to sparse event distributions, the EF-ISF mechanism effectively uncovered clusters and rare event types, as supported by our experimental research results.

Looking forward, we aim to extend this research by integrating supervised learning techniques with the current unsupervised methods to enhance the predictive capabilities of our weighting mechanisms. Such advancements will facilitate not only the segmentation of users based on their browsing behavior but also the anticipation of future actions, thereby offering a proactive approach to user experience optimization on e-commerce platforms.

In conclusion, the incorporation of an EF-ISF based weighting mechanism into data embedding methodologies presents a significant step forward in the nuanced analysis of user navigational behavior, providing a robust foundation for future research and practical applications in the domain of e-commerce analytics.

Author Contributions

Nail Tasgetiren conducted the simulations, analyzed the data, and wrote the initial draft of the manuscript. Mehmet S. Aktas provided academic guidance on the project including in the methodology and system design, provided feedback about the simulation results, and contributed to writing and revising the manuscript. Ilgin Safak managed the project, verified the simulation results, provided critical feedback on the methodology, assisted with data interpretation, and contributed to writing and revising the manuscript.

Acknowledgments

The authors would like to thank Hepsiburada R&D Center and Fibabanka R&D Center for their support.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

Data available on request from the authors.

References

1. A. Aizawa, "An Information-Theoretic Perspective of Tf–Idf Measures," *Information Processing and Management* 39, no. 1 (2003): 45–65, https://doi.org/10.1016/S0306-4573(02)00021-3.

2. K. W. Church, "Word2Vec," *Natural Language Engineering* 23, no. 1 (2017): 155–162, https://doi.org/10.1017/S1351324916000334.

3. A. Grover and J. Leskovec, "Node2vec: Scalable Feature Learning for Networks," in *KDD '16. Association for Computing Machinery* (New York, NY: Association for Computing Machinery (ACM), 2016), 855–864.

4. B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online Learning of Social Representations," in *KDD '14. Association for Computing Machinery* (New York, NY: Association for Computing Machinery (ACM), 2014), 701–710.

5. L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "Struc2vec: Learning Node Representations From Structural Identity," in *KDD '17. Association for Computing Machinery* (New York, NY: Association for Computing Machinery (ACM), 2017), 385–394.

6. H. Nguyen, K. Tran, S. Thomassey, and M. Hamad, "Forecasting and Anomaly Detection Approaches Using LSTM and LSTM Autoencoder Techniques With the Applications in Supply Chain Management," *International Journal of Information Management* 57 (2021): 102282, https://doi.org/10.1016/j.ijinfomgt.2020.102282.

7. M. Ahmed, R. Seraj, and S. M. S. Islam, "The K-Means Algorithm: A Comprehensive Survey and Performance Evaluation," *Electronics* 9, no. 8 (2020): 1295, https://doi.org/10.3390/electronics9081295.

8. D. Deng, "DBSCAN Clustering Algorithm Based on Density," in *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)* (Hefei, China: IEEE, 2020), 949–953, https://doi.org/10.1109/IFEEA51475.2020.00199.

9. P. An, Z. Wang, and C. Zhang, "Ensemble Unsupervised Autoencoders and Gaussian Mixture Model for Cyberattack Detection," *Information Processing and Management* 59, no. 2 (2022): 102844, https://doi.org/10.1016/j.ipm.2021.102844.

10. S. Wang, C. Huang, D. Yu, and X. Chen, "VulGraB: Graph-Embedding-Based Code Vulnerability Detection With Bi-Directional Gated Graph Neural Network," *Software: Practice and Experience* 53, no. 8 (2023): 1631–1658, https://doi.org/10.1002/spe.3205.

11. H. Sankar, V. Subramaniyaswamy, V. Vijayakumar, S. Arun Kumar, R. Logesh, and A. Umamakeswari, "Intelligent Sentiment Analysis Approach Using Edge Computing-Based Deep Learning Technique," *Software: Practice and Experience* 50, no. 5 (2020): 645–657, https://doi.org/10.1002/spe.2687.

12. M. Parimala, R. M. Swarna Priya, M. Praveen Kumar Reddy, C. Lal Chowdhary, R. Kumar Poluru, and S. Khan, "Spatiotemporal-Based Sentiment Analysis on Tweets for Risk Assessment of Event Using Deep Learning Approach," *Software: Practice and Experience* 51, no. 3 (2021): 550–570, https://doi.org/10.1002/spe.2851.

13. P. Jiménez, J. C. Roldán, F. O. Gallego, and R. Corchuelo, "On the Synthesis of Metadata Tags for HTML Files," *Software: Practice and Experience* 50, no. 12 (2020): 2169–2192, https://doi.org/10.1002/spe.2886.

14. H. Li, Y. Qu, Y. Liu, R. Chen, J. Ai, and S. Guo, "Self-Admitted Technical Debt Detection by Learning Its Comprehensive Semantics via Graph Neural Networks," *Software: Practice and Experience* 52, no. 10 (2022): 2152–2176, https://doi.org/10.1002/spe.3117.

15. J. M. T. Wu, Z. Li, G. Srivastava, M. H. Tasi, and J. C. W. Lin, "A Graph-Based Convolutional Neural Network Stock Price Prediction With Leading Indicators," *Software: Practice and Experience* 51, no. 3 (2021): 628–644, https://doi.org/10.1002/spe.2915.

16. Y. Liu and A. Simpson, "Privacy-Preserving Targeted Mobile Advertising: Requirements, Design and a Prototype Implementation," *Software: Practice and Experience* 46, no. 12 (2016): 1657–1684, https://doi.org/10.1002/spe.2403.

17. B. Requena, G. Cassani, J. Tagliabue, C. Greco, and L. Lacasa, "Shopper Intent Prediction From Clickstream E-Commerce Data With Minimal Browsing Information," *Scientific Reports* 10, no. 1 (2020): 16983, https://doi.org/10.1038/s41598-020-73622-y.

18. M. Zavali, E. Lacka, and J. Smedt, "Shopping Hard or Hardly Shopping: Revealing Consumer Segments Using Clickstream Data," *IEEE Transactions on Engineering Management* 70, no. 4 (2023): 1353–1364, https://doi.org/10.1109/TEM.2021.3070069.

19. Y. Ozyurt, T. Hatt, C. Zhang, and S. Feuerriegel, "A Deep Markov Model for Clickstream Analytics in Online Shopping," in *WWW '22. Association for Computing Machinery* (New York, NY: Association for Computing Machinery (ACM), 2022), 3071–3081.

20. M. Grohe, "Word2vec, Node2vec, Graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data," in *PODS'20. Association for Computing Machinery* (New York, NY: Association for Computing Machinery (ACM), 2020), 1–16.

21. D. Cahyani and I. Patasik, "Performance Comparison of TF-IDF and Word2Vec Models for Emotion Text Classification," *Bulletin of Electrical Engineering and Informatics* 10, no. 5 (2021): 2780–2788, https://doi.org/10.11591/eei.v10i5.3157.

22. P. F. Muhammad, R. Kusumaningrum, and A. Wibowo, "Sentiment Analysis Using Word2vec and Long Short-Term Memory (LSTM)

for Indonesian Hotel Reviews," *Procedia Computer Science* 179 (2021): 728–735, https://doi.org/10.1016/j.procs.2021.01.061.

23. J. Jin, M. Heimann, D. Jin, and D. Koutra, "Toward Understanding and Evaluating Structural Node Embeddings," *ACM Transactions on Knowledge Discovery From Data* 16, no. 3 (2021): 1–32, https://doi.org/10.1145/3481639.

24. A. Dehghan-Kooshkghazi, B. Kamiński, U. Kraiński, P. Prałat, and F. Théberge, "Evaluating Node Embeddings of Complex Networks. *Journal of Complex*," *Networks* 10, no. 4 (2022): cnac030, https://doi.org/10.1093/comnet/cnac030.

25. W. Wen, D. D. Zeng, J. Bai, K. Zhao, and Z. Li, "Learning Embeddings Based on Global Structural Similarity in Heterogeneous Networks," *IEEE Intelligent Systems* 36, no. 6 (2021): 13–22, https://doi.org/10.1109/MIS.2020.3027677.

26. L. Wang, C. Huang, Y. Lu, W. Ma, R. Liu, and S. Vosoughi, "Dynamic Structural Role Node Embedding for User Modeling in Evolving Networks," *ACM Transactions on Information Systems* 40, no. 3 (2021): 1–21, https://doi.org/10.1145/3472955.

27. Y. Pei, X. Du, J. Zhang, G. Fletcher, and M. Pechenizkiy, "Struc2gauss: Structural Role Preserving Network Embedding via Gaussian Embedding," *Data Mining and Knowledge Discovery* 34, no. 4 (2020): 1072–1103, https://doi.org/10.1007/s10618-020-00684-x.

28. Z. H. Chen, Z. H. You, Z. H. Guo, H. C. Yi, G. X. Luo, and Y. B. Wang, "Prediction of Drug–Target Interactions From Multi-Molecular Network Based on Deep Walk Embedding Model," *Frontiers in Bioengineering and Biotechnology* 8 (2020): 338, https://doi.org/10.3389/fbioe.2020.00338.

29. K. Berahmand, E. Nasiri, M. Rostami, and S. Forouzandeh, "A Modified DeepWalk Method for Link Prediction in Attributed Social Network," *Computing* 103, no. 10 (2021): 2227–2249, https://doi.org/10.1007/s00607-021-00982-2.

30. H. Yu, R. Ma, J. Chao, and F. Zhang, "An Overlapping Community Detection Approach Based on Deepwalk and Improved Label Propagation," *IEEE Transactions on Computational Social Systems* 10, no. 1 (2023): 311–321, https://doi.org/10.1109/TCSS.2022.3152579.

31. P. K. Padigela and R. Suguna, "A Survey on Analysis of User Behavior on Digital Market by Mining Clickstream Data," in *Proceedings of the Third International Conference on Computational Intelligence and Informatics*, eds. K. S. Raju, A. Govardhan, B. P. Rani, R. Sridevi, and M. R. Murty (Singapore: Springer Singapore, 2020), 535–545.

32. G. B. Martins, J. P. Papa, and H. Adeli, "Deep Learning Techniques for Recommender Systems Based on Collaborative Filtering," *Expert Systems* 37 (2020): e12647, https://doi.org/10.1111/exsy.12647.

33. X. Wang, H. Zhang, W. Huang, and M. R. Scott, "Cross-Batch Memory for Embedding Learning," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 2019 (2020): 6387–6396.

34. Z. Lin, Z. Kang, L. Zhang, and L. Tian, "Multi-View Attributed Graph Clustering," *IEEE Transactions on Knowledge and Data Engineering* 35, no. 2 (2023): 1872–1880, https://doi.org/10.1109/TKDE.2021.3101227.

35. S. Amin, M. I. Uddin, S. Hassan, et al., "Recurrent Neural Networks With TF-IDF Embedding Technique for Detection and Classification in Tweets of Dengue Disease," *IEEE Access* 8 (2020): 131522–131533, https://doi.org/10.1109/ACCESS.2020.3009058.

36. J. Piskorski and G. Jacquet, "TF-IDF Character N-Grams Versus Word Embedding-Based Models for Fine-Grained Event Classification: A Preliminary Study," in *Proceedings of the Workshop on Automated Extraction of Socio-Political Events From News 2020*, eds. A. Hürriyetoğlu, E. Yörük, V. Zavarella, and H. Tanev (Marseille, France: European Language Resources Association (ELRA), 2020), 26–34.

37. M. Seo and K. Y. Lee, "A Graph Embedding Technique for Weighted Graphs Based on LSTM Autoencoders," *Journal of Information Processing Systems* 16, no. 6 (2020): 1407–1423.

38. A. Taheri, K. Gimpel, and T. Berger-Wolf, *Learning Graph Representations With Recurrent Neural Network Autoencoders* (London, UK: KDD Deep Learning Day, 2018).

39. C. Chu, Z. Li, B. Xin, et al., "Deep Graph Embedding for Ranking Optimization in E-Commerce," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy: Association for Computing Machinery (ACM), 2018), 2007–2015.

40. E. Pujastuti, A. D. Laksito, R. Hardi, R. I. Perwira, and A. Arfriandi, "Handling Sparse Rating Matrix for E-Commerce Recommender System Using Hybrid Deep Learning Based on LSTM, SDAE and Latent Factor," *International Journal of Intelligent Engineering and Systems* 15, no. 2 (2022): 379–393.

41. S. Khoshraftar, S. Mahdavi, A. An, Y. Hu, and J. Liu, "Dynamic Graph Embedding via LSTM History Tracking," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (Washington, DC: IEEE, 2019), 119–127.

42. J. Wu, Z. Qiu, Z. Zeng, R. Xiao, I. Rida, and S. Zhang, "Graph Autoencoder Anomaly Detection for E-Commerce Application by Contextual Integrating Contrast With Reconstruction and Complementarity," *IEEE Transactions on Consumer Electronics* 70 (2024): 1623–1630.

43. D. Xu, C. Ruan, E. Körpeoglu, S. Kumar, and K. Achan, "Product Knowledge Graph Embedding for E-Commerce," in *Proceedings of the 13th International Conference on Web Search and Data Mining* (Houston, TX, USA: IEEE, 2019), 672–680.

44. K. Saini and A. Singh, "A Content-Based Recommender System Using Stacked LSTM and an Attention-Based Autoencoder," *Measurement: Sensors* 31 (2023): 100975.

45. K. Bandara, P. Shi, C. Bergmeir, H. Hewamalage, Q. H. Tran, and B. Seaman, "Sales Demand Forecast in E-Commerce Using a Long Short-Term Memory Neural Network Methodology," *ArXiv.2019.abs/1901.04028*.

46. C. Shen, "Encoder Embedding for General Graph and Node Classification," *arXiv preprint arXiv:2405.15473*. 2024.

47. B. Rozemberczki, R. Davies, R. Sarkar, and C. Sutton, "Gemsec: Graph Embedding With Self Clustering," in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (Vancouver, British Columbia, Canada: IEEE, 2019), 65–72.

48. A. Lutov, D. Yang, and P. Cudré-Mauroux, "Bridging the Gap Between Community and Node Representations: Graph Embedding via Community Detection," *IEEE International Conference on Big Data (Big Data)* 2019 (2019): 2681–2690, https://doi.org/10.1109/BigData47090.2019.9006038.

49. B. Kamiński, P. Prałat, and F. Théberge, "A Scalable Unsupervised Framework for Comparing Graph Embeddings," *Algorithms and Models for the Web Graph* 12091 (2020): 52–67.

50. M. Nacar, M. S. Aktas, M. Pierce, et al., "VLab: Collaborative Grid Services and Portals to Support Computational Material Science," *Concurrency and Computation: Practice and Experience* 19, no. 12 (2007): 1717–1728.

51. G. Aydin, A. Sayar, H. Gadgil, et al., "Building and Applying Geographical Information System Grids," *Concurrency and Computation: Practice and Experience* 20, no. 14 (2008): 1653–1695.

52. M. Aktas, G. Aydin, A. Donnellan, et al., "iSERVO: Implementing the International Solid Earth Research Virtual Observatory by Integrating Computational Grid and Geographical Information Web Services," *Computational Earthquake Physics: Simulations, Analysis and Infrastructure* 2 (2007): 2281–2296.

53. G. Fox, M. S. Aktas, G. Aydin, et al., "Algorithms and the Grid," *Computing and Visualization in Science* 12 (2009): 115–124.

54. Y. Uygun, R. F. Oguz, E. Olmezogullari, and M. S. Aktas, "On the Large-Scale Graph Data Processing for User Interface Testing in Big Data

Science Projects," in *2020 IEEE International Conference on Big Data (Big Data)* (Atlanta, GA, USA: IEEE, 2020), 2049–2056.

55. E. Olmezogullari and M. S. Aktas, "Representation of Click-Stream Datasequences for Learning User Navigational Behavior by Using Embeddings," in *2020 IEEE International Conference on Big Data (Big Data)* (Atlanta, GA, USA: IEEE, 2020), 3173–3179.

56. E. Olmezogullari and M. S. Aktas, "Pattern2Vec: Representation of Clickstream Data Sequences for Learning User Navigational Behavior," *Concurrency and Computation: Practice and Experience* 34, no. 9 (2022): e6546.

57. M. Sahinoglu, K. Incki, and M. S. Aktas, "Mobile Application Verification: A Systematic Mapping Study," in *Computational Science and Its Applications–ICCSA 2015: 15th International Conference Proceedings, Part V 15* (Banff, AB: Springer, 2015), 147–163.

58. M. Pierce, G. C. Fox, M. S. Aktas, et al., "The QuakeSim Project: Web Services for Managing Geophysical Data and Applications," *Earthquakes: Simulations, Sources and Tsunamis* 165 (2008): 635–651.

59. M. Kapdan, M. Aktas, and M. Yigit, "On the Structural Code Clone Detection Problem: A Survey and Software Metric-Based Approach," in *Computational Science and Its Applications–ICCSA 2014: 14th International Conference, Proceedings, Part V 14* (Guimarães, Portugal: Springer, 2014), 492–507.

60. I. Kärkkäinen and P. Fränti, "Minimization of the Value of Davies-Bouldin Index," in *Proceedings of the IASTED International Conference on Signal Processing and Communications (SPC'2000)* (Banff, Alberta: IASTED/ACTA Press, 2000), 426–432.

61. L. Agarwal, K. Thakral, G. Bhatt, and A. Mittal, "Authorship Clustering Using TF-IDF Weighted Word-Embeddings," in *Proceedings of the 11th Annual Meeting of the Forum for Information Retrieval Evaluation* (Kolkata, India: Association for Computing Machinery, 2019), 24–29.

62. C. W. Schmidt, "Improving a TF-IDF Weighted Document Vector Embedding," 2019.arXiv preprint arXiv:1902.09875.

63. G. Quattrocchi, D. A. Tamburri, and W. J. Van Den Heuvel, "Making Service Continuity Smarter With Artificial Intelligence: An Approach and Its Evaluation," *Software: Practice and Experience* 53, no. 10 (2023): 1887–1901, https://doi.org/10.1002/spe.3230.

64. N. Taşgetiren, "Clickstream of a E-Commerce Dataset," 2024 accessed October 5, 2024, https://www.kaggle.com/code/nailtasgetiren/clickstream-of-a-e-commerce-dataset.

65. T. Büyüktanır, "Clickstreams of A Coffee Shop Mobile Applications," 2024 accessed October 5, 2024, https://cutt.ly/Lw36wTue.

66. R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling With Large Corpora. ELRA 2010, Valletta," http://is.muni.cz/publication/884893/en.

67. V. Maaten and G. Hinton, "Visualizing Data Using t-SNE," *Journal of Machine Learning Research* 9, no. 11 (2008), 2579–2605.

68. T. A. Caswell, M. Droettboom, A. Lee, et al., "matplotlib/matplotlib: REL: v3. 3.1. Zenodo," 2020, https://doi.org/10.5281/zenodo.3984190.