Data Mining and Knowledge Discovery
https://doi.org/10.1007/s10618-022-00874-9

Category tree distance: a taxonomy-based transaction distance for web user analysis

Yinjia Zhang[1,2] · Qinpei Zhao[1] · Yang Shi[1] · Jiangfeng Li[1,3] · Weixiong Rao[1]

Received: 14 September 2020 / Accepted: 15 September 2022

Abstract

With the emergence of webpage services, huge amounts of customer transaction data are flooded in cyberspace, which are getting more and more useful for profiling users and making recommendations. Since web user transaction data are usually multi-modal, heterogeneous and large-scale, the traditional data analysis methods meet new challenges. One of the challenges is the distance definition on two transaction data or two web users. The distance definition takes an important role in further analysis, such as the cluster analysis or k-nearest neighbor query. We introduce a *category tree distance* in this paper, which makes use of the product taxonomy information to convert the user transaction data to vectors. Then, the similarity between web users can be evaluated by the vectors from their transaction data. The properties of the distance like upper and lower bounds and the complexity analysis are also given in the paper. To investigate the performance of the proposal, we conduct experiments on real web user transaction data. The results show that the proposed distance outperforms the other distances on user transaction analysis.

Responsible editor: Fei Wang.

✉ Qinpei Zhao
qinpeizhao@tongji.edu.cn

Yinjia Zhang
yinjiazhang@aalto.fi

Yang Shi
shiyang@tongji.edu.cn

Jiangfeng Li
lijf@tongji.edu.cn

Weixiong Rao
wxrao@tongji.edu.cn

1 School of Software Engineering, Tongji University, Shanghai, China

2 Department of Computer Science, School of Science, Aalto University, Espoo, Finland

3 Key Laboratory of Blockchain and Cyberspace Governance of Zhejiang Province, Hangzhou, China

Published online: 13 October 2022

Keywords Transaction data · Distance metric · Cluster analysis · k-nearest neighbor query · Tree structure · Taxonomy

1 Introduction

Web user data analysis has attracted great attention these years as the prosperity of the Internet economy. People benefit from the web data analysis in every aspect. For example, transaction records, reviews and rates on e-commerce platforms can be employed in recommendation systems; Check-in data can help to generate Point of Interests (POI) on maps; Trajectories are fundamentals of identifying similar routes and drivers. Learning the proper representations of users, which is usually the first step of the analysis, is vital to the final performance. However, the diversity of the web user data hampers the generation of the representation, making it difficult to extract effective information.

In this paper, we focus on the evaluation of distance between users' transaction data with the help of the taxonomy information. One of the most important issues is how to represent the user's interaction history and the taxonomy information simultaneously, which decides the way to evaluate the relevance among users or between the user and the entity. The traditional method, representing users' data by sets and employ jaccard distance to measure the similarities (Giannotti et al. 2002; Guidotti et al. 2017; Segond and Borgelt 2011; Tummala et al. 2018), is difficult to make use of the hypernym-hyponym information. In Chen et al. (2018), a user's related entities as well as their category information are organized into tree structures and a distance metric between tree structures is proposed. However, there are limitations when using tree structures, such as the inability to calculate centers and inefficiency of computing distance. These years, user embedding techniques which harnesses deep neural networks (Gong et al. 2020; Lee et al. 2019; Liang et al. 2018; Ni et al. 2018) achieve great success. Continuous vectors can represent users' features well. Relevance between users is reflected by the distance between vectors. However, most of the web user embedding models do not take taxonomy information into consideration, and the interpretability of many deep embedding methods is still a problem.

To take an advantage of the taxonomy information to evaluate distance between users' transaction data, we propose a novel method called *category tree distance* for transforming user transaction data to a low dimensional vector in an embedding space by making utilizing the taxonomy information of the products. Compared with existing methods, our proposal has several advantages. First is the distance takes use of tree structures of the taxonomy information. The taxonomy of a set of products can be represented by a tree, in which nodes usually represent categories and descendant nodes are the subcategories represented by their parent nodes. Secondly, Our proposal is flexible. It allows products' category chains (e.g. *Book*→*Fantacy*→*Classic*) in different lengths. Such a length is assumed as a constant for all products in Chen et al. (2018). In real-life, one product may have several different category chains. An electronic toothbrush can be found by <Health & Household→Oral Care→Toothbrushes & Accessories→Powered Toothbrushes→Ultrasonic> and also by <Beauty & Personal Care>. This situation is taken into account in the proposal for flexibility. Thirdly,

the proposal is with low time and space complexity, which are linear to the size of the users and products, while it is more natural to evaluate the similarity of two users.

We further employ the category tree distance in the cluster analysis. Cluster analysis can divide users into groups in which users are similar to each others, which is a widely used method in transaction data mining when labels are not available. In the cluster analysis, distance definition plays a critical role. It affects the number of clusters and the way the clusters are generated. In this paper, we conducted experiments on three real-life datasets. The experimental results show that the category tree distance can reflect the similarity between users' categorical interests effectively and such performance is robust on different datasets.

The rest of the paper is organized as: Sect. 2 gives an overview on the related work. Section 3 presents the details of our proposal. We investigate the distribution of the category tree distance and apply the proposal to the clustering analysis on the datasets from three different websites. The results of the experiments are reported in Sect. 4. At last, conclusions and future work are given in Sect. 5.

2 Related work

Distance between users' transaction data is a critical problem in user transaction data analysis. Since the users can be represented by the transaction data they have consumed, the analysis on the web users leads to the analysis on their transaction data. Based on the representation of the data, the methods to evaluate the similarity between users' transactional interests can be classified into two kinds: vector based method and non-vector based methods.

The idea of the vector based methods is that firstly embed the transaction data to a vector space and then take the distance between the feature vectors. In this kind, the transformation from the transaction data to the vectors is critical to the final performance. Some researchers employ graph embedding to make this transformation. Models, like DeepWalk (Perozzi et al. 2014), Node2Vec (Grover and Leskovec 2016) and LINE (Tang et al. 2015), take networks as graphs and learn low dimensional vectors of users with the help of skip-gram (Mikolov et al. 2013) or random walk. Some other methods focus on the sequence mining. Nguyen et al. (2018) propose a method to learn low-dimensional vectors for transactions based on the frequent itemsets. In Ni et al. (2018), authors propose a model to capture the universal user representations from multiple e-commerce tasks using user behavior sequences. Okura et al. (2017) generate user representations by using a recurrent neural network with users' browsing histories as input sequences. However, most of them ignore the extra information such as the taxonomy information, which is the abstract of the transaction products and can help to discover potential common interests.

A majority of the taxonomy related researches focus on the construction of the hierarchy structure or the embedding of the taxonomy. Although these methods do not embed users' transaction data directly, they are still inspiring to our task. Some researches construct the taxonomy with the help of statistic and topic mining models. Blei et al. (2003) propose hierarchical latent Dirichlet allocation(HLDA) to learn hierarchical topics from data using Bayesian approach. Liu et al. (2012) make use

of Bayesian Rose Tree to tackle the problem of deriving a taxonomy from a set of keywords with the help of additional knowledge and contexts. Some other researchers propose graph based methods to solve the problem. Kang et al. (2016) propose TaxoFinder, which uses key concepts retrieved from domain corpus to build a graph and then generates a taxonomy using a graph analytic algorithm. Construction of the taxonomy by clustering is also a popular kind of methods. TaxoGen (Zhang et al. 2018) employs spherical k-means clustering (Dhillon and Modha 2001) to divide topic words into sub groups hierarchically to generate a taxonomy. For taxonomy embedding, graph embedding methods like spectral embedding (Shi and Malik 2000) are popular among the researchers. Recently, Nickel and Kiela (2017) extended the taxonomy embedding from the Euclidean space to the hyperbolic space, which can naturally reflect the hierarchy by the space's negative curvature.

The non-vector based methods directly measure the distance between users' transaction data based on the data structures like sets and trees. Commonly, the transaction data are categorical. For categorical data, the simplest comparison measure is overlap (Ienco et al. 2012). Therefore, the Jaccard distance is usually considered in the transaction data analysis. Tummala et al. (2018) propose a transaction clustering method that employs the Jaccard distance as a similarity measure. Some researches (Giannotti et al. 2002; Guidotti et al. 2017) make efforts on modifying the Jaccard distance in classic clustering algorithms such as the K-Means. Giannotti et al. (2002) adapt the distance definition in K-Means to the Jaccard distance and the centers of clusters are redefined. *Txmeans* (Guidotti et al. 2017) focuses on clustering massive individual transaction datasets, where a new clustering algorithm combines the Jaccard distance and a variation of the *xmeans* (Ramadan and Tairi 2015) together. However, the Jaccard distance suffers from high computation cost and high sparsity on transaction data. Besides, the Jaccard distance ignores the taxonomy information of the products. A user's transaction data as well as the category information is a pruned tree of the entire taxonomy. To calculate the distance between two trees, tree-based distance like the tree edit distance (McVicar et al. 2016; Valiente 2001), the PQ-gram tree distance (Augsten et al. 2008) and the binary branch tree distance (Yang et al. 2005) can be employed.

Like string edit distance, tree edit distance is to calculate the cost of converting a tree to another by operations such as insertion, deletion and substitution. There are many methods to compute tree edit distance. For example, Zhang and Shasha (1989) proposes a method which traverses two trees from the root node to the leaf node. And the method in Valiente (2001) traverses trees in the inverse direction. The PQ-gram distance decomposes the original tree to a set of small trees named pq-grams which have P non-leaf nodes and Q leaf nodes. The pq-gram tree distance between two trees is the symmetric difference between their pq-gram sets. The binary-branch tree distance (Yang et al. 2005) converts trees to their binary tree representations. The labels of binary-branches are collected. A tree structure vector, of which each element is the occurrences of a certain binary-branch's labels, is generated for each original tree. The binary-branch tree distance between two trees is the L_1 distance between their tree structure vectors. All these tree distances can evaluate the similarity between trees. However, for transaction data, due to the enormous amount of items and the tiny portion which a user may consume, these methods often suffer from the high sparsity.

In order to capture the semantic meaning of the tree hierarchy, which is ignored by tree edit distances, a PurTree distance is proposed in Chen et al. (2018). It combines the Jaccard distance and taxonomy information together. However, PurTree distance assumes that all leaf nodes in a purchase tree have equal depth, which means all products must be organized with a fixed number of category levels. It is not so general in transaction data analysis. Cho and Kim (2004) propose a method to convert a web user's transaction interests to a vector with the help of taxonomy information. Then, a modification of their method is proposed in Albadvi and Shahbazi (2009). However, there is an implicit assumption of such works that children of a non-leaf node must be in the same kind, either all non-leaf nodes or all leaf nodes.

3 Methodology

In this section, we introduce a distance, named *Category Tree Distance*, to evaluate the similarity of users' interests through their transaction data. The proposal makes use of the taxonomy information of the products to profile the users' transaction data and further evaluates the distance between users based on their profilings. The properties of the proposal are reported and some toy examples are given for the better illustration.

3.1 Preliminary concepts

Before giving the definition of the category tree distance, we clarify some basic concepts in Table 1 to make the presentation more understandable.

Taking Amazon as an example, the domain is the entire set of the products of the Amazon. There are a lot of categories on the Amazon such as *Books*, *Cloth* and *Electronics* and the taxonomy of the domain reflects the hyponymy relationships among these categories. Another example is the Yelp, a website of businesses on local services. People can rate those businesses or publish related reviews. A product on the Yelp can be a restaurant or a cinema. The category path of a restaurant may be like $\{Restaurants, Chinese, Szechuan\}$. The user transaction data involved in this paper is defined in Definition 1.

Definition 1 (User Transaction Data) The **transaction data** of the user u on the domain Θ is defined as $TD(u) = \{\langle p_i, f_i\rangle | 1 \leq i \leq n, p_i \in \Theta, f_i \geq 1\}$, where p_i represents a product and f_i represents the times of the user u interacting with the product p_i. For convenience, denote the product set of the $TD(u)$ as $TD_p(u)$ and the iteraction frequency of the $TD_p(u)$ as $TD_f(u)$.

3.2 Category tree distance

More definitions on the introduced *Category Tree Distance* are given. The distance evaluates the categorical similarity between two users' transaction data from two steps: profiling the transaction data by vectors and obtaining the distance by these vectors.

We profile a user's transaction data with the help of the corresponding category information. Given the taxonomy of a domain, the root node represents the domain

Table 1 The preliminary concepts

Concept	Description
Product	The entities on e-commerce platforms
Domain	The entire set of the products which belong to a specific platform, such as all the products of the Amazon
Category	A group of products which consists a subset of the domain. All products in a category share a common specific property. For example, if the domain is the entire set of the products on an e-commerce platform such as Amazon, one category can be *Book*. A category may have its sub-categories. Category *Book* may have a sub-category named *Science Fiction*. The common property of the products in the *Science Fiction* is more specific than that of the *Book*
Category path	An ordered list $\rho = [c_1, c_2, \ldots, c_l]$, $(l \geq 1)$ which represents the category information of product. The c_1 is the most general category and the c_l is the most specific category. In this paper, we name c_1 as the **top category** of the ordered list and c_l as the **end category**. Note that a product may have more than one category path
Taxonomy	The hierarchical category structure of the domain. It is a tree structure of which the root node represents all the products of the domain and internal nodes are categories. The hyponymy relation between two categories is represented by a parent-child relation in the tree structure

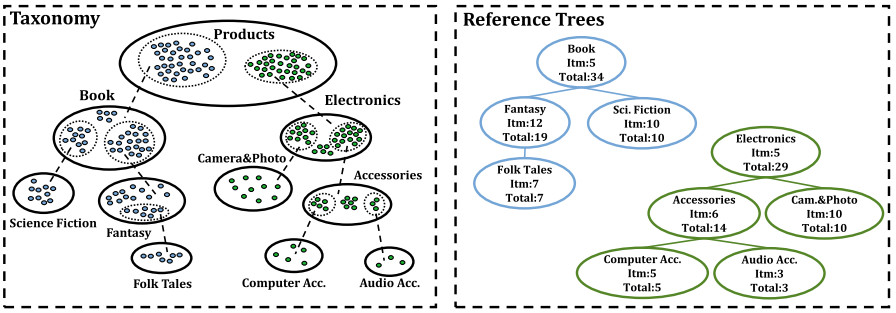


Fig. 1 An example on the definitions of the taxonomy and reference trees

itself and a child of the root node represents a specific category of the entire domain. A such category is named as a **top category** as mentioned before. Recursively, a top category as well as its descendants compose a sub-taxonomy. The idea of the profiling is to evaluate the matching between a user's transaction history and all such sub-taxonomies separately to construct a feature vector. For each sub-taxonomy, we build a new tree structure named *Reference Tree*. The details of the reference tree are shown by the Definition 2.

Definition 2 (Reference Tree) A reference tree is a tree structure $T(V, E, R)$ with a vertice set V, a directed edges set E and a root node $R \in V$. An edge $\langle u, v \rangle$ begins from node $u \in V$, ends at $v \in V$ and u is the only one parent of v. R represents a top category. Every node $v \in V$ contains the information as follows:

1. Name of the category represented by v.
2. $chd(v)$: the number of child nodes of v.
3. $pdt(v)$: the number of products which have a category path whose end category is represented by v. If v is a leaf node, then $pdt(v) > 0$.
4. $total(v)$: the sum of the pdt values of the v as well as its descendants.

Figure 1 gives an example of the product taxonomy and its two reference trees. The left side shows the tree taxonomy whose domain is a set of product. There are two top categories *Electronics* and *Book*. The right side gives the two reference trees of the top categories. The *Electronics*, as a top category, has two sub-categories, which makes $chd(Electronics) = 2$. There are totally 29 products in the *Electronics* sub-taxonomy so the $total(Electronics) = 29$. The value of $pdt(Electronics)$ is 5 because there are 5 products in the database which contain the category path $\{Electronics\}$. In this case, $\{Electronics\}$ is also an end category. For node *Accessories*, its *chd* is 2 because it has two sub-categories *Computer Accessories* and *Audio Accessories*. The $pdt(Accessories)$ is 6 means there are 6 products containing the category path $\{Electronics, Accessories\}$.

The reference trees play very important roles in the profiling step. In this paper, we use a vector, of which each dimension is the relevance between the user's transaction data and a reference tree, to profile the user's interest on the domain. To measure the user's relevance to a reference tree, we define the relevance between the user's

transaction data(Definition 1) and an arbitrary node inside the reference tree. Then take the sum of all nodes' relevance as the transaction data's relevance to the tree.

Definition 3 (Relevance Between User Transaction Data and a Node) Given a node v_l at the l-th level of a reference tree, its category path is $\{v_1, \ldots, v_l\}$. The user u's transaction data is $TD(u) = \{\langle p_i, f_i\rangle \mid 1 \leq i \leq n\}$, let $TD(u, v_l) = \{\langle p, f\rangle \mid p \in v_l, \langle p, f\rangle \in TD(u)\}$ denote all the products the u bought from the category tied to the v_l. The relevance between $TD(u)$ and the node v_l is:

$$rel(TD(u), v_l) = \prod_{i=1}^{l} [\frac{A_l(v_i)}{total(v_i)}]^{\frac{1}{i}} \cdot B(TD(u, v_l)) \tag{1}$$

where $A_l(v_i)$ is defined as:

$$A_l(v_i) = \begin{cases} total(v_{i+1}) & i < l \\ pdt(v_i) & i = l \end{cases} \tag{2}$$

and $B(TD(u, v_l))$ is defined as:

$$B(TD(u, v_l)) = \begin{cases} 0 & pdt(v_l) = 0 \\ \frac{\sum_{f \in TD_f(u, v_l)} f}{pdt(v_l) - |TD(u, v_l)| + \sum_{f \in TD_f(u, v_l)} f} & pdt(v_l) \neq 0 \end{cases} \tag{3}$$

Definition 3 gives the definition of the relevance between a user's transaction data $TD(u)$ and a reference tree node v_l. The $\prod_{i=1}^{l} [\frac{A_l(v_i)}{total(v_i)}]^{\frac{1}{i}}$ part of the Equation (1) is to describe an accumulation process of the user's interest on a category v_l through a reference tree. Starting from the top category, the user goes through the sub-categories and finally arrives at the target node v_l. At each ancestral node $v_i (1 \leq i < l)$, the user chooses the v_{i+1} as the next node with the probability $\frac{total(v_{i+1})}{total(v_i)}$. When arriving the last node v_l, the choice is to stay at the current node instead of going to a child, with the probability of $\frac{pdt(v_l)}{total(v_i)}$. Since the choice narrows down the user's interest on the base of the previous choices, we multiply the probabilities of all choices to reflect the chance of the user's transactional interest going to the node v_l, which is treated as the relevance between the transaction data and the reference tree in this paper.

However, with the increment of the node level, the distinction between the node and its parent becomes more and more implicit. For example, the difference between the *Computer Accessories* and the *Accessories* is less than that between the *Accessories* and *Electronics*. The decline of the difference makes the narrowing down effect of the current choice not as obvious as that of the previous one. In this paper, we use the i-th root of the $\frac{total(v_{i+1})}{total(v_i)}$ to reflect the decline of the difference, where i is the level of the node v_i. Due to $\frac{total(v_{i+1})}{total(v_i)} \leq 1$, a high level leads to the root being more close to 1, which reduces the above-mentioned narrowing down effect.

From a certain reference tree node, the more types of products the user u interacts or the more times the u interacts a certain product for, the higher relevance should be.

These two motivations are reflected by the Equation (3). We give the proof of them in the next subsection.

With the relevance $rel(TD(u), v_l)$, the relevance between a user's transaction data and a reference tree T is defined as:

Definition 4 (Relevance Between User Transaction Data and a Reference Tree) Given a reference tree $T(R, V, E)$, the relevance between a user's transaction data $TD(u)$ and the reference tree T is:

$$Rel(TD(u), T) = \sum_{v \in V} rel(TD(u), v) \tag{4}$$

With the relevances between the user transaction data and all of the reference trees, the category tree vector of a certain user u is obtained.

Definition 5 (Category Tree Vector) Given a set of reference trees $\{T_i(R_i, V_i, E_i) \mid 1 \leq i \leq D\}$ and a user u's transaction data $TD(u)$, the category tree vector of the $TD(u)$ is defined as:

$$CTV(TD(u)) = [\, w_1 \cdot Rel(TD(u), T_1),\ \ldots, w_d \cdot Rel(TD(u),\ T_d)\,] \tag{5}$$

where $w_i = \frac{total(R_i)}{\sum_{j=1}^{D} total(R_j)}$.

When the number of products of a reference tree T_i is much larger than that of other trees, the value of $Rel(TD(u), T_i)$ often be much smaller than that of the other trees. So we use the weight w to reduce the effect of the imbalance of the product quantities.

With the above approach, the user u's transaction data $TD(u)$ is converted into the category tree vector first. Then the distance between two users' transaction data is evaluated by their category tree vectors. In this paper, cosine distance is taken to evaluate the distance between two category tree vectors.

Definition 6 (Category Tree Distance) Given two users u_1 and u_2 with transaction data sets $TD(u_1)$ and $TD(u_2)$. The category tree distance between $TD(u_1)$ and $TD(u_2)$ is:

$$d_{cate}(TD(u_1), TD(u_2)) = 1 - \frac{CTV(TD(u_1)) \cdot CTV(TD(u_2))}{\|CTV(TD(u_1))\|_2 \|CTV(TD(u_2))\|_2}. \tag{6}$$

3.3 Properties

In this subsection, we give the properties possessed by the category tree vector and distance. First we prove that the relevance in Definition 2 increases with the increment of the product types and the frequency of the interaction. Then, the upper and lower bounds of this relevance are shown and proved.

Theorem 1 *Given a reference tree $T(R, V, E)$ as well as two users u_1 and u_2. The users' transaction data sets are $TD(u_1) = \{\langle p_i, f_i\rangle \| 1 \leq i \leq t\}$ and $TD(u_2) = TD(u_1) \cup \{\langle p_{t+1}, f_{t+1}\rangle\}$. Then:*

$$Rel(TD(u_1), T) \leq Rel(TD(u_2), T). \tag{7}$$

Theorem 1 According to the definitions of the the category tree vectors, we get:

$$\begin{aligned} &Rel(TD(u_2), T) - Rel(TD(u_1), T) \\ &= \sum_{v_l \in V} rel(TD(u_2), v_l) - rel(TD(u_1), v_l) \\ &= \sum_{v_l \in V} \prod_{i=1}^{l} [\frac{A_l(v_i)}{total(v_i)}]^{\frac{1}{l}} [B(TD(u_2, v_l)) - B(TD(u_1, v_l))], \end{aligned}$$

The $\prod_{i=1}^{l} [\frac{A_l(v_i)}{total(v_i)}]^{\frac{1}{l}}$ part is always greater than zero. For the $B(TD(u_2, v_l)) - B(TD(u_1, v_l))$ part, we get:

$$\begin{aligned} &B(TD(u_2, v_l)) - B(TD(u_1, v_l)) \\ &= \frac{f_{t+1}(pdt(v_l) - \mid TD(u_1, v_l) \mid) + \sum_{i=1}^{t} f_i(\mid TD(u_2, v_l) \mid - \mid TD(u_1, v_l) \mid)}{(pdt(v_l) - \mid TD(u_2, v_l) \mid + \sum_{f \in TD_f(u_2, v_l)} f)(pdt(v_l) - \mid TD(u_1, v_l) \mid + \sum_{f \in TD_f(u_1, v_l)} f)} \end{aligned}$$

Because $TD(u_1) \subset TD(u_2)$, so $|TD(u_2, v_l)| - |TD(u_1, v_l)| \geq 0$. Thus it can be seen that $B(TD(u_2, v_l)) - B(TD(u_1, v_l)) \geq 0$, and further the $Rel(TD(u_2), T) - Rel(TD(u_1), T) \geq 0$ holds. □

Theorem 2 *Given a reference tree $T(R, V, E)$ as well as two users u_1 and u_2. The users' transaction data sets are $TD(u_1) = \{\langle p_i, f_i\rangle \| 1 \leq i \leq t\}$ and $TD(u_2) = \{TD(u_1) - \{\langle p_t, f_t\rangle\}\} \cup \{\langle p_t, f_t + \beta\rangle \mid \beta \geq 1\}$. Then:*

$$Rel(TD(u_1), T) \leq Rel(TD(u_2), T). \tag{8}$$

Theorem 2 The proof of the Theorem 2 is similar to that of the Theorem 1. The difference is on the $B(TD(u_2, v_l)) - B(TD(u_1, v_l))$ part. This time we get:

$$\begin{aligned} &B(TD(u_2, v_l)) - B(TD(u_1, v_l)) \\ &= \frac{\beta(pdt(v_l) - \mid TD(u_2, v_l) \mid)}{(pdt(v_l) - \mid TD(u_2, v_l) \mid + \sum_{f \in TD_f(u_2, v_l)} f)(pdt(v_l) - \mid TD(u_1, v_l) \mid + \sum_{f \in TD_f(u_1, v_l)} f)} \\ &\geq 0. \end{aligned}$$

So the $Rel(TD(u_2), T) - Rel(TD(u_1), T) \geq 0$ holds. □

The Theorem 1 and Theorem 2 indicate that a user's interest on a category grows with the product types and interaction frequency. However, such growth will not continue forever. There exists bounds of the relevance $Rel(TD(u), T)$. We give $Rel(TD(u), T)$'s upper and lower bounds as follows.

Theorem 3 *Given a reference tree $T(R, V, E)$ and a user u's transaction data $TD(u)$, $Rel(TD(u), T) \in [0, |V|]$.*

Theorem 3 First we consider the relevance between $TD(u)$ and an arbitrary reference tree node $v_l \in V$. For the $TD(u, v_l)$ there are three conditions as follows:

1. When the user buys nothing from the v_l, the $TD(u, v_l)$ is an empty set, leading to the $B(TD(u, v_l)) = 0$ and further $rel(TD(u), v_l) = 0$.
2. According to the Theorem 1 and the Theorem 2, the $rel(TD(u), v_l)$ gets larger while the user interacts more products or purchases a product for more times. Especially when the times of buying the products become very large, the $B(TD(u, v_l))$ will be very close to 1 and thus make the $rel(TD(u), v_l)$ very close to $\prod_{i=1}^{l}[\frac{A_l(v_i)}{total(v_i)}]^{\frac{1}{l}}$.
3. When the user has bought all the products from the v_l, the $B(TD(u, v_l))$ is 1 and $rel(TD(u), v_l) = \prod_{i=1}^{l}[\frac{A_l(v_i)}{total(v_i)}]^{\frac{1}{l}}$.

It can be concluded from the above conditions that the $rel(TD(u), v_l) \leq \prod_{i=1}^{l}[\frac{A_l(v_i)}{total(v_i)}]^{\frac{1}{l}}$ holds all the time. This limit can be further relaxed to 1 and we get $rel(TD(u), v_l) \in [0, 1]$. So we get $Rel(TD(u), T) \in [0, |V|]$. □

3.4 Complexity analysis

In this subsection we discuss the space and time complexities of the proposal. We use P to denote the entire set of the products. A product $p \in P$ can mostly have t category paths and the longest category path is l. There are D top categories in the taxonomy of the domain. The reference tree of the i-th top category is noted by T_i and $|T_i|$ is the number of the nodes in the T_i. The user set in the database is U and the largest size of users' transaction data is m. We divide the proposal into three steps: building the category trees, mapping the users' transaction data to vectors and calculating the distance between two users' vectors.The complexity analysis of these three parts are as follows:

1. **Build reference trees**. This step is conducted only once during the whole analysis. For each top category, a reference tree node is initialized as the root node of the corresponding reference tree. Its pdt and $total$ are both zeros. Then we traverse the category paths of all the products to construct the D reference trees. For each category path, we insert it to the related tree and update the pdt and $total$ values while inserting. Thus the construction of the D reference trees can be done in one traverse of all products' category paths, of which the time complexity is $O(tl|P|)$. Because there is no extra space needed during the construction except for the storage of the products' category paths as well as D reference trees, the space complexity of this step is $O(tl|P| + \sum_{i=1}^{D}|T_i|)$.
2. **Map user transaction data to vectors**. We analyze the complexity of this step from the view of the category paths which appear in the user u's transaction data. For a node v, if the $TD(u, v) = \emptyset$, then the relevance must be zero. Thus we only need to calculate the relevance with the nodes from which the user interacts with the items, and all these nodes are end categories of the user's category paths. So

the mapping can be done in one traverse of u's category paths. Therefore the time complexity of the step is $O(mtl|U|)$. Since there is no need for extra storage space, the space complexity is also $O(mtl|U|)$.

3. **Calcuate distance between vectors**. In this step, we calculate the cosine distance between two users. The time and space complexities are both $O(D)$.

3.5 Toy example

To better present the theory of the category tree distance, we give a toy example in Fig. 2. The example is composed of five parts: 1. Products and Category Paths; 2. User Transaction Data; 3. Relevance with Reference Tree Node; 4. Relevance with Reference Tree; 5. Category Tree Vector and Distance.

In the toy example, we show the transaction data of three users. All steps of the distance evaluation are shown in details. The $User\#1$ buys a SD card from the *Electronics* and the book *The Lord of the Rings*. $User\#2$ buys two books, *Foundation* and *The Lord of the Rings*, from the category *Book* but nothing from the *Electronics*. The transaction data of the $User\#3$ is quite similar to that of the $User\#1$, but the $User\#3$ buys the SD card for three times. Intuitively, the $User\#1$ and $User\#3$ should be close to each other and both of them are not that close to the $User\#2$. And the distance between $User\#2$ and $User\#3$ should be larger than that between $User\#2$ and $User\#1$ because it seems that the $User\#3$ is more interested in the *Electronics* than the $User\#1$ does, in which the $User\#2$ is totally not interested. In the Fig. 2, the category tree distances between the three users agree with such intuitions completely, which shows the effectiveness of our proposal.

4 Experiments

In this section, we present the performance of the category tree distance for web user transaction data. The proposal is compared with other existing distances or similarity measures on real datasets. We also show the performance of the distance in clustering analysis, where users calculated as similar by the distance are grouped.

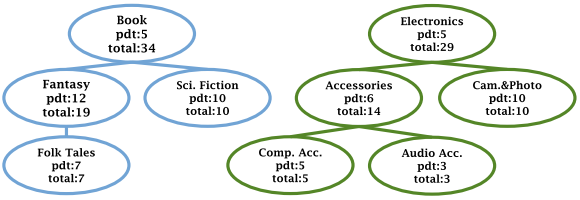
4.1 Experimental settings

All the experiments are conducted on a computer with an Intel i7-8750h processor and 32GB memory. All the distances in the experiments are implemented in Java. For the clustering analysis , we employ the implementation of *scikit-learn* in Python.

4.1.1 Real-life transaction datasets

In this paper, we use three real-life transaction datasets to investigate the performance of the category tree distance.

Fig. 2 A toy example of the category tree distance



Products and Category Paths

Product	Category Paths
San Disk SD Card	{ Electronics, Accessories, Comp. Acc. }, { Electronics, Cam.&Photo }
The Lord of the Ring	{ Book, Fantasy }
Foundation	{ Book, Sci. Fiction }

User Transaction Data

User#1 : SanDisk SD Card × 1 + The Lord of the Rings × 1
User#2 : Foundation × 1 + The Lord of the Rings × 1
User#3 : SanDisk SD Card × 3 + The Lord of the Rings × 1

Relevance with Reference Tree Node

$$rel(TD(User\#1), <Comp.Acc.>) = \left(\frac{14}{29}\right)^{\frac{1}{1}}\left(\frac{5}{14}\right)^{\frac{1}{2}}\left(\frac{5}{5}\right)^{\frac{1}{3}}\frac{1}{5-1+1} = 0.0577$$

$$rel(TD(User\#1), <Cam.\&Photo>) = \left(\frac{10}{29}\right)^{\frac{1}{1}}\left(\frac{10}{10}\right)^{\frac{1}{2}}\frac{1}{10-1+1} = 0.0345$$

SanDisk SD Card

$$rel(TD(User\#1), <Fantasy>) = \left(\frac{19}{34}\right)^{\frac{1}{1}}\left(\frac{12}{19}\right)^{\frac{1}{2}}\frac{1}{12-1+1} = 0.03700$$

The Lord of the Ring

$$rel(TD(User\#2), <Sci.Fiction>) = \left(\frac{10}{34}\right)^{\frac{1}{1}}\left(\frac{10}{10}\right)^{\frac{1}{2}}\frac{1}{10-1+1} = 0.0294$$

Foundation

$$rel(TD(User\#2), <Fantasy>) = \left(\frac{19}{34}\right)^{\frac{1}{1}}\left(\frac{12}{19}\right)^{\frac{1}{2}}\frac{1}{12-1+1} = 0.03700$$

The Lord of the Ring

$$rel(TD(User\#3), <Comp.Acc.>) = \left(\frac{14}{29}\right)^{\frac{1}{1}}\left(\frac{5}{14}\right)^{\frac{1}{2}}\left(\frac{5}{5}\right)^{\frac{1}{3}}\frac{3}{5-1+3} = 0.1236$$

$$rel(TD(User\#3), <Cam.\&Photo>) = \left(\frac{10}{29}\right)^{\frac{1}{1}}\left(\frac{10}{10}\right)^{\frac{1}{2}}\frac{3}{10-1+3} = 0.0862$$

SanDisk SD Card

$$rel(TD(User\#3), <Fantasy>) = \left(\frac{19}{34}\right)^{\frac{1}{1}}\left(\frac{12}{19}\right)^{\frac{1}{2}}\frac{1}{12-1+1} = 0.03700$$

The Lord of the Ring

Relevance with Reference Tree

$Rel(TD(User\#1), [Book]) = rel(TD(User\#1), <Fantasy>) = 0.0370;$
$Rel(TD(User\#1), [Electronics])$
$= rel(TD(User\#1), <Comp.Acc.>) + rel(TD(User\#1), <Cam.\&Photo>) = 0.0922;$
$Rel(TD(User\#2), [Book])$
$= rel(TD(User\#2), <Sci.Fiction>) + rel(TD(User\#2), <Fantasy>) = 0.0664;$
$Rel(TD(User\#2), [Electronics]) = 0;$
$Rel(TD(User\#3), [Book]) = rel(TD(User\#3), <Fantasy>) = 0.0370;$
$Rel(TD(User\#3), [Electronics])$
$= rel(TD(User\#3), <Comp.Acc.>) + rel(TD(User\#3), <Cam.\&Photo>) = 0.2099$

Category Tree Vector and Distance

$$CTV(User\#1) = \left(Rel(TD(User\#1), [Book]) \times \frac{34}{63}, \quad Rel(TD(User\#1), [Electronics]) \times \frac{29}{63}\right)$$
$$= (0.0200, \quad 0.0424)$$

$$CTV(User\#2) = \left(Rel(TD(User\#2), [Book]) \times \frac{34}{63}, \quad Rel(TD(User\#2), [Electronics]) \times \frac{29}{63}\right)$$
$$= (0.0358, \quad 0.0000)$$

$$CTV(User\#3) = \left(Rel(TD(User\#3), [Book]) \times \frac{34}{63}, \quad Rel(TD(User\#3), [Electronics]) \times \frac{29}{63}\right)$$
$$= (0.0200, \quad 0.0966)$$

Distance:

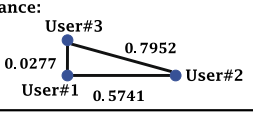


Table 2 The summary of the real-life datasets

Real-life dataset	Products	Users	Top Cate.	Min\Max\Avg. depth
Amazon Recom. dataset	9,354,832	21,176,523	85	1\9\3.1
Yelp Challenge dataset	174,567	1,326,101	22	2\4\3.0
Gowalla Dataset	2,724,891	319,063	7	2\3\2.9

1. **The Yelp challenge dataset**[1] (Yelp). The dataset is released by Yelp, a business recommendation and crowd-sourced review platform. It is composed of business information, user information and user activities' records including reviews, tips, photos and check-in. All the businesses are organized into 22 top categories,[2] such as restaurant, food and hotel.
2. **Amazon Recommendation Dataset** (He and McAuley 2016; McAuley et al. 2015) (Amazon). The dataset consists of 142.8 million reviews of 9 million products. All these reviews are produced by 21.1 million users. The entire set of products are categorized into 85 top categories,[3] for example, clothing, kitchen. Some top categories are further divided into sub-categories but some others are not.
3. **Gowalla Dataset** (Liu et al. 2013, 2014)(Gowalla). This dataset was collected from Gowalla which is a popular location-based social network. The dataset includes the user profiles, user friendship, location profiles, and users check-in history made before June 1, 2011. Toltally there are 36,001,959 check-ins made by 319,063 users over 2,844,076 locations. The locations in Gowalla are grouped into 7 main categories, e.g., Food, Nightlife, Outdoors, Shopping and Travel, and each main category consists of several subcategories.

Table 2 summarizes the basic characteristics of the three real-life datasets. Besides the number of the products and users, we also present the number of the top categories(Top Cate.). These top categories and their descendants compose the reference trees of the proposal. The column Min\Max\Avg. Depth in the table shows the minimum, maximum and average depth of the top categories of each datasets.

The scale of the Amazon is the largest among the three datasets, with more than 9 million products and 21 million users. It contains 85 top categories which is the most among the three datasets. However, the average depths of the top categories in three datasets are very close to each other, which indicates that in the Amazon dataset, there are many top categories without sub-categories.

The Gowalla contains about 3 million locations but only about 319 thousand users. Although there are only 7 top categories, the average depth of these top categories is very close to the maximum depth. It is because that the depths of most top categories are 3, which means the reference tree structure is more balanced compared to that of the other two datasets.

[1] https://www.yelp.com/dataset/challenge.

[2] https://www.yelp.com/developers/documentation/v3/category_list.

[3] http://jmcauley.ucsd.edu/data/amazon/.

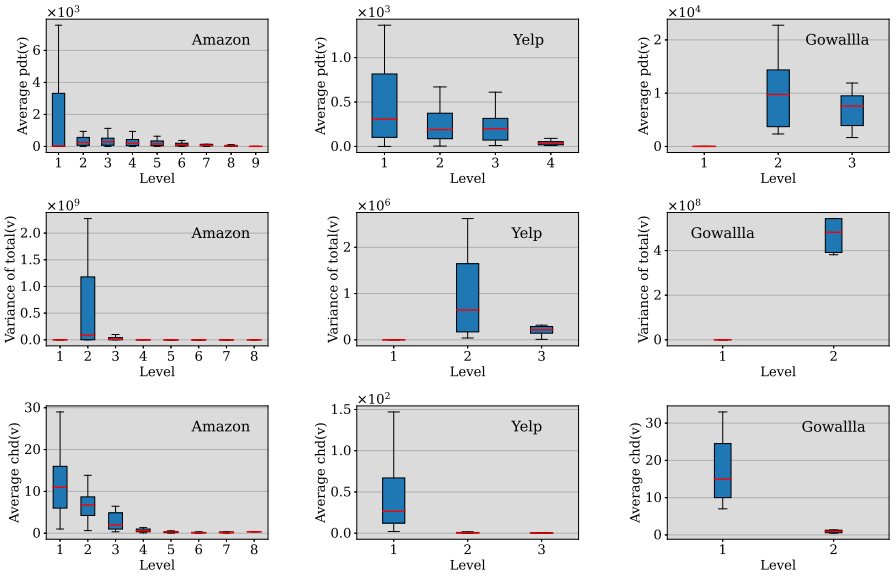


Fig. 3 The box plot of average pdt, average chd and variance of the $total$ at each level of reference trees from different datasets (Amazon, Yelp and Gowalla). The x-axis is the depth level and the y-axis is the statistics values of different reference trees in terms of different levels

The Yelp is the one with the simplest taxonomy structure, as it has the least products but 22 top categories, which indicates the pdt values of the reference tree nodes are less than those of other two datasets.

According to the definitions, the value of the category tree distance is influenced by the pdt, $total$ values of the nodes. Besides, the chd value of a node reflects the structural complexity of the tree and indirectly affects the $total$ values of its child nodes.

Thus, to comprehensively show the characteristics of different datasets, we provide the statistics of each level of reference trees from the datasets. The nodes' average pdt values, average chd values and the variance of the $total$ values are calculated. The average pdt values reflect the products distribution at different tree levels. The average chd values reflect the structural complexity of the trees and the variance of the $total$ reflects the degree of the imbalance of the product numbers among the child nodes. The detailed statistics are shown in Fig. 3, where box plots of the statistics are demonstrated at each depth level of the reference trees.

From the Fig. 3, it can be concluded that more products locate at lower depth levels on both Amazon and Yelp datasets, however, the majority of the products locate at the higher depth levels of its reference tree on the Gowalla dataset. On all the three datasets, the number of child nodes decreases with the increment of the tree level, which means the tree structure near the root node is more complicated than that near the leaf nodes. From the variance of $total$, the second levels of the reference trees are the most unbalanced level. The total product numbers of the top category's different sub-categories are quite different to each other.

4.1.2 Baselines

In the experiment, we evaluate the performance of the category tree distance(**cate**) against the following methods:

1. **AKM Distance(akm)** (Munthe Caspersen et al. 2017) is a distance measure between the classes in a hierarchical label structure for classification problems. We implement the measure to evaluate the distance between category nodes in a taxonomy.
2. **Poincare Embedding(poincare)** (Nickel and Kiela 2017) is a representation learning method which embeds the hierarchical symbolic data into hyperbolic space to capture hierarchy and similarity. We take use of this method to learn the representations of the nodes in a taxonomy. Then, the distance measure in the hyperbolic space is employed to evaluate the distance between the category nodes.
3. **Spectral Embedding(spec)** (Shi and Malik 2000) is a classic method which embeds graphs to vector spaces. In this paper we constructed the adjacency matrix of the taxonomy tree first and then employ the spectral embedding on the matrix. The RBF kernal is used while constructing the weight matrix.
4. **PQ-Grams Distance(pqgram)** (Augsten et al. 2008) is a pattern based distance measure between tree structures. The method captures the structure similarity between trees by counting the same local structures. In this paper, we construct a pruned tree for each user, which only keeps the user-related categories and takes products as leaf nodes, to present the user's transaction data and then applied the PQ-gram to obtain the distance.
5. **Binary Branch Distance(binary)** (Yang et al. 2005) is a tree based distance. The distance first converts trees to their left-child right-sibling representations and then collects the labels of the binary branches in the new representations. The distance between two trees is the difference between their branch labels. We employ the same scheme as the PQ-Gram's to obtain the distance between users' transaction data.
6. **Jaccard Distance(jaccard)** (Levandowsky and Winter 1971) is a measure to evaluate the distance between two sets by $1 - \frac{A \cap B}{A \cup B}$. In this paper, we use the products' sets to represent the users' transaction data.

The original AKM distance, Poincare embedding and Spectral embedding are intended for evaluating the distance between nodes of a taxonomy. In our experiments, they are adapted in the following strategy to get the distance between users' transaction data:

Given the transaction data of the user u as $TD(u) = \{\langle p_1, f_1\rangle >, \ldots, \langle p_n, f_n\rangle\}$, construct a bucket C_u of all the categories related to the $TD(u)$. The C_u contains all the categories which appear in the category paths of the products in $TD(u)$. Further, if a product $p \in TD_p(u)$ is purchased for f times, its categories will repeat for f times in C_u; if a category is related to both $p_i \in TD_p(u)$ and $p_j \in TD_p(u)$, it will repeat for at least $f_i + f_j$ times in C_u. For two user transaction data on users u_1 and

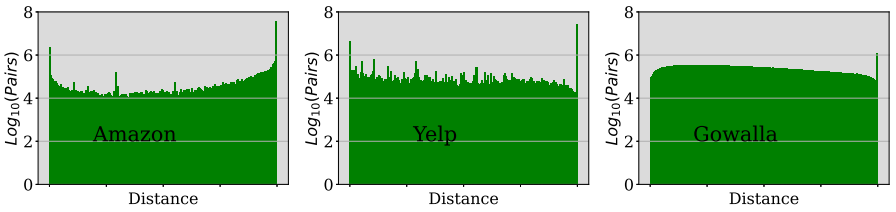


Fig. 4 The distance distributions of the category tree distance on 10,000 users from Amazon, Yelp and Gowalla datasets. The x-axis contains 200 bins between the minimum and maximum distance values. The y-axis is the log_{10} of user pairs in each distance scale range

u_2, the akm, poincare and the spec distances are calculated according to:

$$d_a(TD(u_1), TD(u_2)) = \begin{cases} 0 & TD(u_1) = TD(u_2) \\ \frac{\sum_{c_1 \in C_{u_1}} \sum_{c_2 \in C_{u_2}} d'(c_1, c_2)}{|C_{u_1}||C_{u_2}|} & TD(u_1) \neq TD(u_2) \end{cases}$$

where $d'(c_1, c_2)$ is calculated by the AKM distance, the hyperbolic distance between poincare embeddings or the Euclidean distance between spectral embeddings. In the actual implementation, if the $C(u_1)$ or the $C(u_2)$ are empty sets, the adapted distance d_a will be assigned a very large number instead of the infinity.

4.2 Distribution

To investigate the distribution of the category tree distance, we randomly choose 10,000 users, who interact with at least one product from each dataset. Seven kinds of distances are calculated between each pair of users. For each distance on each dataset, the range from the minimum to the maximum are divided into 200 bins. For each bin, the number of the user pairs whose distance locates in the bin is counted. Because the users are picked randomly from the original datasets, ideally we expect the user pair numbers in different bins are in the same scale, and the similarities of the randomly selected users lead to a nearly uniform distribution.

The distributions of the category tree distance on the users from three datasets are shown in Fig. 4. The x-axis contains 200 bins between the the minimum and maximum distance values. The y-axis is the log_{10} of the user pairs which locates in each bin. It can be seen that except the last bin, the user pair numbers of the majority of the bins are in the same scale on all the three datasets. In details, the histogram on the Gowalla dataset is the trimmest one. The Amazon's distribution comes to the second place and the Yelp is the last. This order is the same as the order of the product numbers per user of the three datasets. It indicates that the more products users buy, the better the category tree distance can reflect their similarities.

In order to compare the proposal with the six other distances, we conducted the same experiments. Figure 5 shows the distributions of the six distance measures on the Amazon. Compared with the category tree distance, the distributions of the akm distance, spec and poincare show similarly as the proposal. The performances of the two tree-based distance methods and the jaccard distance are very umbalanced. Most

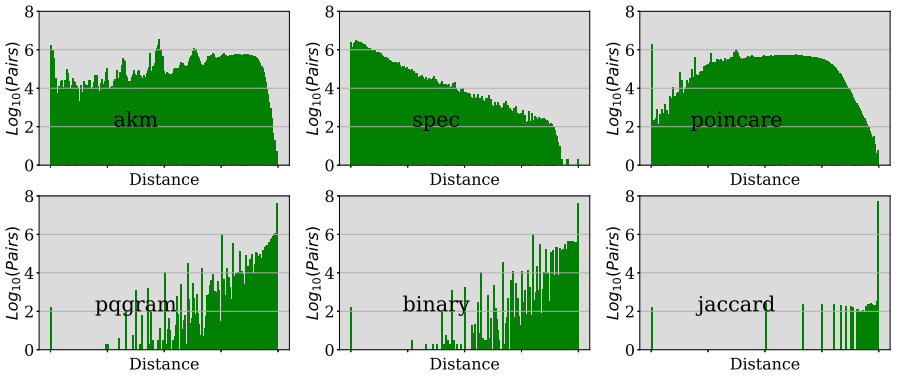


Fig. 5 The distance distributions of the six baselines on 10,000 users from Amazon Recommendation Datasets. The x-axis is the distance values and the y-axis is the log_{10} of user pairs in each distance scale range

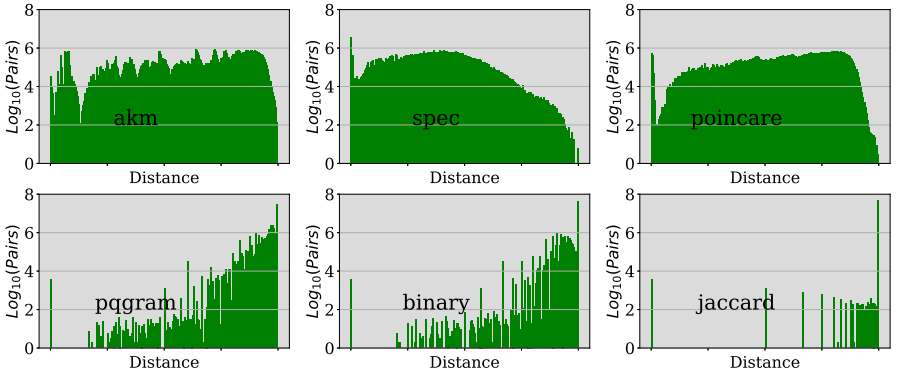


Fig. 6 The distance distributions of the six baselines on 10,000 users from Yelp Challenge Datasets. The x-axis is the distance values and the y-axis is the log_{10} of user pairs in each distance scale range

distance values locates in the left region, which means that these three measures think two users being far away from each other more often and can not reflct the similarities between users well enough. In other words, the pqgram, binary and the jaccard fail to find similar users.

Figures 6 and 7 give the distributions of the six baselines on the Yelp Challenge dataset and Gowalla dataset respectively. Comparing the results of different distance measures to that of the proposal, we can draw conclusions which are similar to those on the Amazon Recommendation dataset. Further, from the view of the distribution shape, the proposal's shapes on the three datasets are very close to each other, achieving the highest stability comparing with the other methods.

To further investigate the performance of the different distance measures, we randomly pick 50 users from the 10,000 selected users and querying their 200 nearest neighbors. Figure 8 is the curve of the average distance from the queried users to their neighbors. The x-axis represents the rank of the 200 nearest neighbors and the y-axis

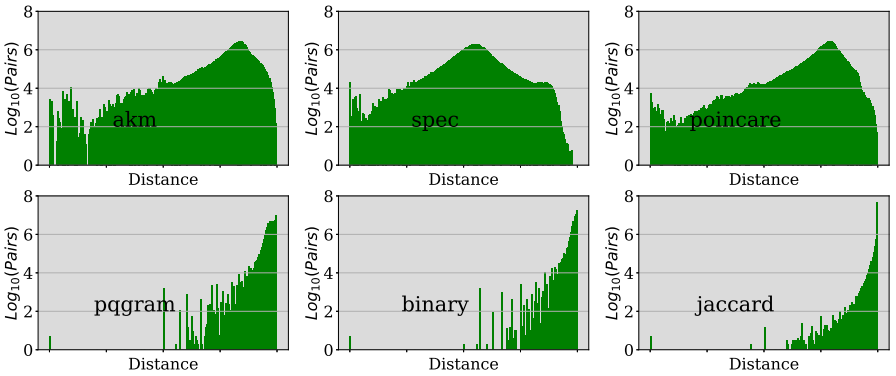


Fig. 7 The distance distributions of the six baselines on 10,000 users from Gowalla Datasets. The x-axis is the distance values and the y-axis is the log_{10} of user pairs in each distance scale range

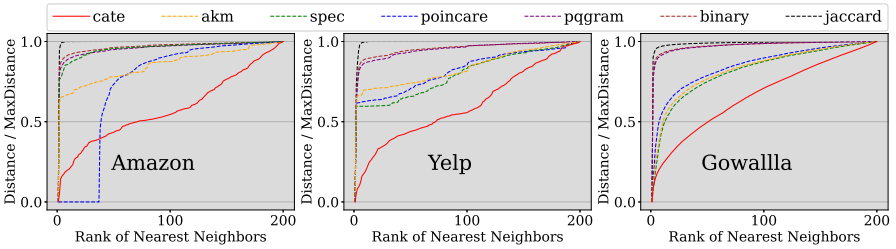


Fig. 8 Average distance curve of 50 users' 200 nearest neighbors. The x-axis is the 200 nearest neighbors and y-axis is the ratio betweem *average distance to the corresponding neighbor* and the *maximum average distance*

is the ratio between the *average distance to the corresponding neighbor* and *maximum average distance* (i.e. the average distance to the 200-th nearest neighbor).

Ideally, We expect there are obvious discrimination between distances to different neighbors. It can be seen that the gradient of the category tree distance's curve is more stable than those of the other methods. A stable gradient means that the distance measure can reflect the difference between adjacent nearest neighbors more effectively. For jaccard distance, the distances from the queried user to the 30-th nearest neighbor and to the 200-th nearest neighbor are equal on all the three datasets, which means we can only query 30 neighbors of a user by the jaccard distance; for poincare distance on the Amazon, the distances from the 1-st nearest neighbor to the 40-th nearest neighbor are totally equal to each other, which means we can not decide who is most similar to the queried user.

4.3 Category tree distance in clustering

In this section, the distances are tested in the clustering analysis. The distance definition plays a critical role in the clustering analysis, affecting the final performance a lot. A well defined distance definition can help the clustering analysis reveal the potential patterns of the data effectively.

Table 3 The summary of the clustering datasets

Dataset	Size	Clusters	P/U[1]	C/U[2]	Avg. CPL[3]
amazon-s1	3000	1000, 600, 600, 800	7.6	12.0	2.8
amazon-s2	3000	1000, 50, 550, 1400	7.1	14.6	2.7
amazon-s3	3000	2000, 200, 300, 500	6.9	9.4	2.4
amazon-s4	4200	900 800 750 600 400 300 200 150 100	7.5	10.5	2.9
yelp-s1	3000	1000, 600, 600, 800	1.3	5.4	2.2
yelp-s2	3000	1000, 50, 550, 1400	1.4	5.6	2.2
yelp-s3	3000	2000, 200, 300, 500	1.2	5.4	2.1
yelp-s4	4200	900 800 750 600 400 300 200 150 100	1.4	5.4	2.2
gowalla-s1	3000	1000, 600, 600, 800	40.1	73.2	2.4
gowalla-s2	3000	1000, 50, 550, 1400	18.9	31.5	2.5
gowalla-s3	3000	2000, 200, 300, 500	24.7	45.8	2.5
gowalla-s4	1900	200, 400, 100, 300, 200, 400, 300	39.2	83.0	2.3

[1] Number of products per user.
[2] Number of category paths per user.
[3] Average category path length

In this part, we fix the clustering algorithm and adopt different distance definitions. Since not all compared distance is able to evaluate the mean value (centroid), which is required by many clustering algorithms, only clustering algorithms that do not require mean or average values are chosen. The spectral clustering, DBSCAN and hierarchical clustering are three typical algorithms of this kind. Through the experiments, it is found that the performance from the agglomerative hierarchical clustering is not comparable to others in terms of its computation cost for a relatively large dataset. As for the DBSACN, the parameter tuning is complex especially when a certain number of cluster is required. Therefore, we take the spectral clustering in the experiments for comparing different distances.

There is few benchmark dataset available for web user clustering or query based on their transactions. Therefore, we extract experimental data from the three datasets in Table 2. We manually generate twelve datasets with labels, four datasets from each real-life dataset, by the following rules:

1. All products in the transaction records of users appear in at least two top categories.
2. Users in the same cluster have the same most-frequently-purchased top category.
3. Use n_1 to denote the number of products which belong to a user u's the most consumed category and n_2 to denote that number of u's the second most consumed category, the n_2/n_1 must be under the 0.5.

The information of the generated datasets are summarized in Table 3. The *P/U* represents the *Number of products Per User*; the *C/U* represents the *Number of Category Paths Per User*; the *Avg. CPL* represents the *Average Category Path Length*.

The **-s1* datasets are the simplest compared to the **-s2*, **-s3* and **-s4* datasets, with balanced clusters. The **-s2* datasets are generated to test the ability of detecting the small clusters in the data. The **-s3* datasets are to test the performance of clustering

when there exists large cluster. And the **-s4* datasets are the most complicated ones, which have more clusters and various cluster sizes.

When ground truth labels exist, external validity index can be used. It measures the matching degree between two sets of clustering labels. In this paper, we select adjusted rand index and normalized mutual information as external indices. They are further discussed as follows:

1. **Adjusted rand index**(ARI) (Hubert and Arabie 1985) compares two clusterings based on pair counting. The index ranges in [0, 1]. A higher value means two clustering results match each other better.
2. **Normalized mutual information**(NMI) (Estevez et al. 2009) compares two clusterings in the view of information entropy. The NMI ranges from 0 to 1. A high value means two sets matching well.

To evaluate if a clustering result matches the ground truth well, one is to check the external index value calculated, the other is to check if the correct number of clusters can be found. Since the performance of the clustering result relies on the distance definition quite much, the evaluation on the clustering result indirectly reflects the performance of the distances.

The ARI and NMI values from the spectral clustering with different distance definitions and different number of clusters (K) on **-s1* dataset are shown by Fig. 9. The **-s1* datasets have four clusters ($K = 4$), which can be determined by the ARI and NMI curves in terms of their maximum values. The cate, akm and spec are the three best methods, of which the performances are much better than that of the other four distance measures. In details, the cate distance has similar performance with the akm and the spec on the amazon-s1 and gowalla-s1, where the ARI and NMI values reach the maximum and are comparable when $K = 4$. However, the clustering results with the spec and the poincare show the highest matching when $K = 5$ on the yelp-s1, while the results with the cate and the akm have the highest values when $K = 4$. The performance of the poincare, pqgram, binary and jaccard is much worse, especially the jaccard.

It can be observed from Fig. 9 that the ARI and NMI have similar performance. Therefore, we only show the NMI values in the following experiments. The clustering results on **-s2*, **-s3* and **-s4* are shown in Fig. 10. It can be concluded from the results on the **-s1* to **-s4* that it is more difficult for the clustering algorithm to discover the data structure when the complexity of the data gets higher.

On the **-s2* datasets, the cate, akm and the spec reach their peaks when $K = 3$ mostly. The only exception is the spec on the amazon-s2. The reason on the performance is that **-s2* datasets are unbalanced, in which there is a small cluster only composed of 50 users. The **-s3* datasets have a relatively large cluster, which leads to clustering results differently. As the yelp has the lowest P/U^1 and C/U^2 (see Table 3), the clustering algorithm performs better on the yelp-s3 than the other two datasets. The clustering algorithms with the cate, akm and poincare clearly discover the data structure. On the gowalla-s3 dataset, the cate shows the best performance with the K determined as 4, meanwhile the NMI value is the highest when $K = 4$.

As the most complicated data sets, **-s4* datasets are the most challenging. On amazon-s4, the maximum values of the NMI are found when $K = 7, 6, 8$ with the

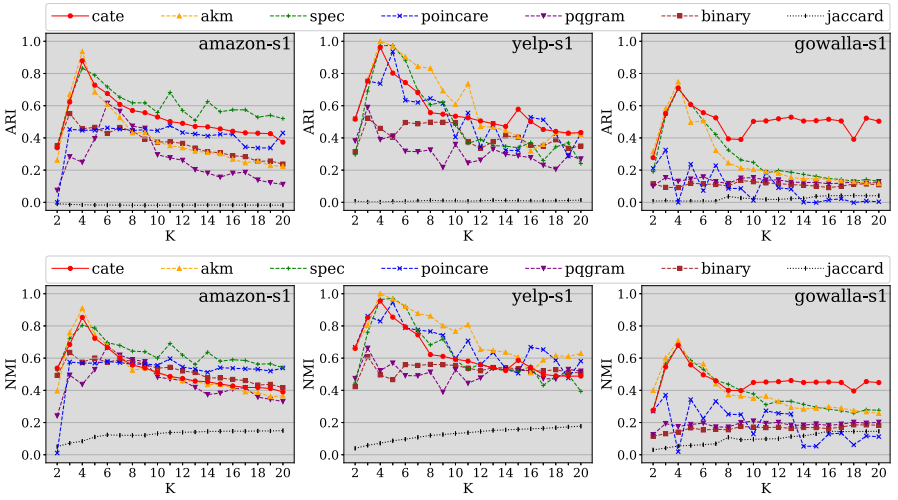


Fig. 9 The adjusted rand index(ARI) and the normalized mutual information(NMI) on different number of clusters (K) from the spectral clustering on the amazon-s1, yelp-s1 and gowalla-s1

cate, akm and spec, which are closed to the ground truth $K = 9$. On yelp-s4 dataset, only the clustering algorithms with the cate and spec are able to discover the data structure. On gowalla-s4, the cate outperforms the other methods. In general, the clustering algorithm with the cate distance has better capability on the **-s4* datasets.

To show the significance of the performance, we take NMI values from the spectral clustering employing the cate, akm, spec and poincare, which are the four best baselines in the clustering task, and further calculate means and standard deviations of the NMI differences between the cate and the other three baselines. The results are shown in Table 4. A positive mean value indicates that the cate is better. Among all the 12 datasets, cate outperforms akm and spec in 6 datasets and poincare in 8 datasets in terms of μ. Besides, the scales of standard deviation in the table are small and close to each other, which indicates the significance of the results. The table shows that the performance of the cate in clustering tasks is at least as good as those of the akm and spec and outperforms all the other baselines. Later, we will show that the cate has great superiority on time efficiency.

We show the distance matrix of the clustering results on the gowalla-s3 in Fig. 11. The users are sorted by their labels generated from the clustering results, and the lighter color means lower distance and darker color represents higher distance. From the figures, we can see the cluster structures discovered by the clustering algorithm employing different distances. The matrix with the cate distance has clearly four clusters, which indicates the superiority of our proposal.

We also record the computation cost of calculating the affinity matrices using different distance measures on dataset amazon-s1 in Table 5. There are two steps to generate the affinity matrix, which are the initialization and calculating the matrix. For the cate, akm, pq-gram and binary distances, the initialization time is the time of constructing the tree structures of users and taxonomies; for the poincare, the initialization time is

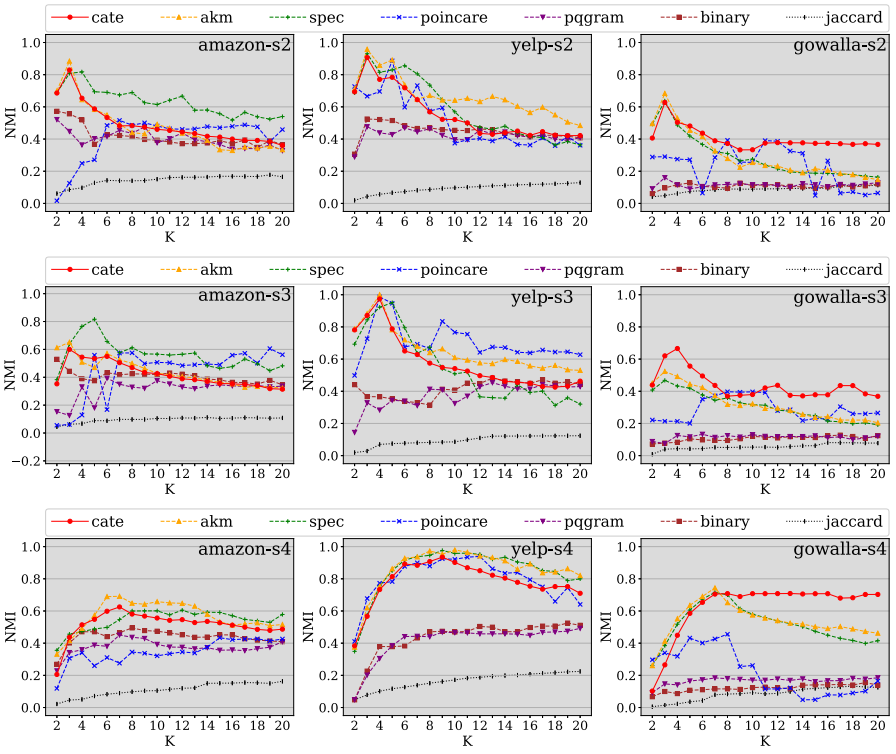


Fig. 10 The normalized mutual information(NMI) on different number of clusters (K) from the spectral clustering on the amazon-s2/s3/s4, yelp-s2/s3/s4 and gowalla-s2/s3/s4

Table 4 The mean(μ) and standard deviation(σ) of NMI's difference between the cate and the three best baselines in all datasets

Datasets	Akm (μ/σ)	Spec (μ/σ)	Poincare (μ/σ)
amazon-s1	**0.009/0.042**	−0.104/0.005	**0.005/0.0164**
amazon-s2	**0.016/0.034**	−0.144/0.059	**0.08/0.244**
amazon-s3	−0.019/0.062	−0.14/0.058	−0.021/0.234
amazon-s4	−0.052/0.044	−0.029/0.059	**0.169/0.086**
yelp-s1	−0.104/0.077	**0.007/0.008**	−0.047/0.076
yelp-s2	−0.112/0.063	−0.033/0.068	**0.042/0.081**
yelp-s3	−0.075/0.042	**0.031/0.085**	−0.127/0.138
yelp-s4	−0.074/0.038	−0.067/0.046	−0.016/0.049
gowalla-s1	**0.074/0.095**	**0.077/0.082**	**0.268/0.137**
gowalla-s2	**0.105/0.090**	**0.112/0.082**	**0.171/0.131**
gowalla-s3	**0.115/0.057**	**0.130/0.063**	**0.149/0.130**
gowalla-s4	**0.089/0.131**	**0.112/0.143**	**0.407/0.252**
$\mu \geq 0$ #/total	**6/12**	**6/12**	**8/12**

Highlight in bold where the cate outperforms its competitors on the mean of NMI's differences

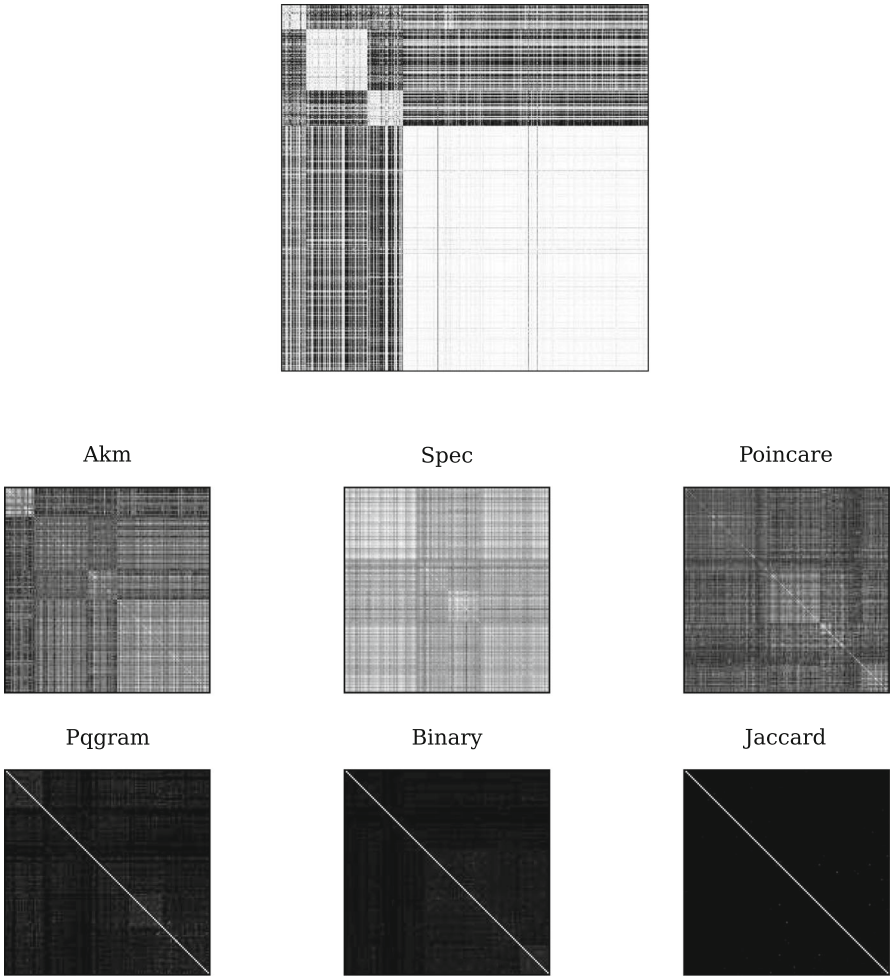


Fig. 11 The category tree distance matrix of the clustering result on gowalla-s3. The lighter color means lower distance and darker color represents higher distance

the time of the training and the training is conducted by a Nvidia GTX 1050 graphic card; for the spec, it is the time of embedding taxonomy nodes to vector spaces.

The cate outperforms the other methods a lot on the efficiency. For the proposed distance, it takes the least time for the affinity matrix calculation because the distance between two users' transaction data is just the distance between two vectors. The spec, poincare and akm distance need to follow the strategy in Sect. 4.1.2, thus they need more time to calculate the matrix. The two tree distance measures, pq-gram distance and binary branch distance, take the longest time to generate the matrix, because for each pair of users, the two methods need to traverse two tree structures to evaluate the distance.

Table 5 Time(ms) of generating affinity matrix of **-s1* Datasets

Dataset	Step	cate	akm	spec	poincare	pqgram	binary	jaccard
amazon-s1	Init.[1]	499	61	405,331	9,871,081	282	282	55
	Mat.[2]	377	86,148	61,876	178,908	466,292	217,103	4850
yelp-s1	Init.	130	11	908	188784	24	24	12
	Mat.	123	24,712	8,775	29,640	204,601	83,900	1402
gowalla-s1	Init.	503	228	149	46011	531	531	214
	Mat.	74	291,321	72,750	187,059	965,997	676,482	33,806

[1]Initialization.
[2]Calculate affinity matrix

5 Conclusion

The analysis on web user transaction data shows huge potential in different areas. As a key problem in the transaction data analysis, the distance definition of two transaction data is a problem should be studied. In this paper, we present a distance so-called *category tree distance* for evaluating the similarity of web users in the aspect of their transaction data with taxonomy information. A *reference tree* is introduced to record taxonomy of products. Based on the reference trees, the category tree distance is to calculate the matching degrees between a transaction tree and reference trees. Then, the matching values are embedded into vectors. The vector of a transaction data can be used to evaluate the similarity of two users in terms of their transaction data. The upper and lower bounds are also given and proven in the paper.

The proposed distance has been compared with different representative distance measures in three aspects: distance distribution, clustering analysis and time efficiency. The distance distribution of the proposal on random datasets is more uniform and stable than those of the baselines. In clustering analysis, the category tree distance's performance is among the top class. In the time efficiency experiment, category tree distance outperforms others with huge superiority. It is at least 97 times faster than akm and 38 times faster than spectral embedding. These experimental results on the real datasets demonstrate the superiority and the stability of the category tree distance.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant Nos. 61972286, 62172301 and 61702371), the Shanghai Municipal Science and Technology Major Project (2021SHZDZX0100), the Science and Technology Program of Shanghai, China (Grant Nos. 20ZR1460500, 22511104300), the Fundamental Research Funds for the Central Universities (No. 22120210545).

References

Albadvi A, Shahbazi M (2009) A hybrid recommendation technique based on product category attributes. Expert Syst Appl 36(9):11,480-11,488. https://doi.org/10.1016/j.eswa.2009.03.046

Augsten N, Böhlen M, Gamper J (2008) The *pq*-gram distance between ordered labeled trees. ACM Trans Database Syst 10(1145/1670243):1670247

Blei DM, Jordan MI, Griffiths TL, et al (2003) Hierarchical topic models and the nested Chinese restaurant process. In Proceedings of the 16th international conference on neural information processing systems. MIT Press, Cambridge, MA, USA, NIPS'03, pp 17–24

Chen X, Fang Y, Yang M et al (2018) Purtreeclust: a clustering algorithm for customer segmentation from massive customer transaction data. IEEE Trans Knowl Data Eng 30(3):559–572. https://doi.org/10.1109/TKDE.2017.2763620

Cho YH, Kim JK (2004) Application of web usage mining and product taxonomy to collaborative recommendations in e-commerce. Expert Syst Appl 26(2):233–246. https://doi.org/10.1016/S0957-4174(03)00138-6

Dhillon IS, Modha DS (2001) Concept decompositions for large sparse text data using clustering. Mach Learn 42(1):143–175. https://doi.org/10.1023/A:1007612920971

Estevez PA, Tesmer M, Perez CA et al (2009) Normalized mutual information feature selection. IEEE Trans Neural Netw 20(2):189–201. https://doi.org/10.1109/TNN.2008.2005601

Giannotti F, Gozzi C, Manco G (2002) Clustering transactional data. In: Proceedings of the 6th European conference on principles of data mining and knowledge discovery. Springer-Verlag, Berlin, Heidelberg, PKDD '02, pp 175–187

Gong L, Lin L, Song W et al (2020) JNET: learning User Representations via joint network embedding and topic embedding. Association for Computing Machinery, New York, NY, USA, pp 205–213

Grover A, Leskovec J (2016) Node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery, New York, NY, USA, KDD '16, pp 855–864. https://doi.org/10.1145/2939672.2939754

Guidotti R, Monreale A, Nanni M, et al (2017) Clustering individual transactional data for masses of users. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery, New York, NY, USA, KDD '17, pp 195–204.https://doi.org/10.1145/3097983.3098034

He R, McAuley J (2016) Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proceedings of the 25th international conference on world wide web. In: International world wide web conferences steering committee, Republic and Canton of Geneva, CHE, WWW '16, pp 507–517. https://doi.org/10.1145/2872427.2883037

Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193–218. https://doi.org/10.1007/BF01908075

Ienco D, Pensa RG, Meo R (2012) From context to distance: learning dissimilarity for categorical data clustering. ACM Trans Knowl Discov Data 10(1145/2133360):2133361

Kang YB, Haghigh PD, Burstein F (2016) Taxofinder: a graph-based approach for taxonomy learning. IEEE Trans Knowl Data Eng 28(2):524–536. https://doi.org/10.1109/TKDE.2015.2475759

Lee H, Im J, Jang S, et al (2019) Melu: Meta-learned user preference estimator for cold-start recommendation. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. Association for Computing Machinery, New York, NY, USA, KDD '19, pp 1073–1082. https://doi.org/10.1145/3292500.3330859

Levandowsky M, Winter D (1971) Distance between sets. Nature 234(5323):34–35. https://doi.org/10.1038/234034a0

Liang, S Zhang, X, Ren Z, Kanoulas E (2018) Dynamic embeddings for user profiling in twitter. Association for Computing Machinery, New York, NY, USA, pp 1764–1773

Liu X, Song Y, Liu S, et al (2012) Automatic taxonomy construction from keywords. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery, New York, NY, USA, KDD '12, pp 1433–1441. https://doi.org/10.1145/2339530.2339754

Liu X, Liu Y, Aberer K, et al (2013) Personalized point-of-interest recommendation by mining users' preference transition. In: Proceedings of the 22nd ACM international conference on information & knowledge management. Association for Computing Machinery, New York, NY, USA, CIKM '13, pp 733–738. https://doi.org/10.1145/2505515.2505639

Liu Y, Wei W, Sun A, et al (2014) Exploiting geographical neighborhood characteristics for location recommendation. In: Proceedings of the 23rd ACM international conference on conference on information and knowledge management. Association for Computing Machinery, New York, NY, USA, CIKM '14, pp 739–748. https://doi.org/10.1145/2661829.2662002

McAuley J, Targett C, Shi Q, et al (2015) Image-based recommendations on styles and substitutes. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval. Association for Computing Machinery, New York, NY, USA, SIGIR '15, pp 43–52. https://doi.org/10.1145/2766462.2767755

McVicar M, Sach B, Mesnage C et al (2016) Sumoted: an intuitive edit distance between rooted unordered uniquely-labelled trees. Pattern Recognit Lett 79:52–59. https://doi.org/10.1016/j.patrec.2016.04.012

Mikolov T, Sutskever I, Chen K et al (2013) Distributed representations of words and phrases and their compositionality. In: Burges C, Bottou L, Welling M et al (eds) Advances in Neural Information Processing Systems, vol 26. Curran Associates, Inc

Munthe Caspersen K, Bjeldbak Madsen M, Berre Eriksen A, et al (2017) A hierarchical tree distance measure for classification. In: Proceedings of the 6th international conference on pattern recognition applications and methods - ICPRAM,, INSTICC. SciTePress, pp 502–509. https://doi.org/10.5220/0006198505020509

Nguyen D, Nguyen TD, Luo W et al (2018) Trans2vec: Learning transaction embedding via items and frequent itemsets. In: Phung D, Tseng VS, Webb GI et al (eds) Advances in Knowledge Discovery and Data Mining. Springer International Publishing, Cham, pp 361–372

Ni Y, Ou D, Liu S, et al (2018) Perceive your users in depth: learning universal user representations from multiple e-commerce tasks. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. Association for Computing Machinery, New York, NY, USA, KDD '18, pp 596–605. https://doi.org/10.1145/3219819.3219828

Nickel M, Kiela D (2017) Poincaré embeddings for learning hierarchical representations. In: Proceedings of the 31st international conference on neural information processing systems. Curran Associates Inc., Red Hook, NY, USA, NIPS'17, pp 6341–6350

Okura S, Tagami Y, Ono S, et al (2017) Embedding-based news recommendation for millions of users. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery, New York, NY, USA, KDD '17, pp 1933–1942. https://doi.org/10.1145/3097983.3098108

Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery, New York, NY, USA, KDD '14, pp 701–710. https://doi.org/10.1145/2623330.2623732

Ramadan H, Tairi H (2015) Collaborative xmeans-em clustering for automatic detection and segmentation of moving objects in video. In: 2015 IEEE/ACS 12th international conference of computer systems and applications (AICCSA), pp 1–2. https://doi.org/10.1109/AICCSA.2015.7507148

Segond M, Borgelt C (2011) Item set mining based on cover similarity. In: Proceedings of the 15th Pacific-Asia conference on advances in knowledge discovery and data mining - Volume Part II. Springer-Verlag, Berlin, Heidelberg, PAKDD'11, pp 493–505

Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905. https://doi.org/10.1109/34.868688

Tang J, Qu M, Wang M, et al (2015) Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, WWW '15, pp 1067–1077. https://doi.org/10.1145/2736277.2741093

Tummala K, Oswald C, Sivaselvan B (2018) A frequent and rare itemset mining approach to transaction clustering. In: Sharma RSM (eds). Data Science Analytics and Applications. Springer Singapore, Singapore, pp 8–18

Valiente G (2001) An efficient bottom-up distance between trees. In: Proceedings eighth symposium on string processing and information retrieval, pp 212–219. https://doi.org/10.1109/SPIRE.2001.989761

Yang R, Kalnis P, Tung AKH (2005) Similarity evaluation on tree-structured data. In: Proceedings of the 2005 ACM SIGMOD international conference on management of data. Association for Computing Machinery, New York, NY, USA, SIGMOD '05, pp 754–765. https://doi.org/10.1145/1066157.1066243

Zhang K, Shasha D (1989) Simple fast algorithms for the editing distance between trees and related problems. SIAM J Comput 18(6):1245–1262. https://doi.org/10.1137/0218082

Zhang C, Tao F, Chen X, et al (2018) Taxogen: unsupervised topic taxonomy construction by adaptive term embedding and clustering. In: Proceedings of the 24th ACM SIGKDD international conference on

knowledge discovery & data mining. Association for Computing Machinery, New York, NY, USA, KDD '18, pp 2701–2709. https://doi.org/10.1145/3219819.3220064