Generator of Synthetic Datasets for Hierarchical Sequential Pattern Mining Evaluation

Michal Šebek, Jaroslav Zendulka

Department of Information Systems
Faculty of Information Technology
Brno University of Technology Technology
Brno, Czech Republic
Email: {isebek,zendulka}@fit.vutbr.cz

***Abstract*—Evaluation is an important part of an algorithm design. Algorithms are typically evaluated on real-world and synthetic datasets. Real-world datasets are appropriate for an evaluation of algorithm properties in practice but it is not easy to make changes only on a selected statistic property of the dataset, e.g. number of input items. In contrast, generated synthetic dataset simply allows changing any of statistic property of the dataset with keeping all others statistic properties. In the paper, we present a procedure for generation of sequence databases with taxonomies for an evaluation of hierarchical sequential pattern mining algorithms.**

***Index Terms*—Sequence pattern mining, synthetic dataset generators, taxonomy.**

I. Introduction

The problem of the generating synthetic dataset is closely linked with the problem of data mining. Typically, when a new data mining problem is formulated, scientists need to verify their algorithms or methods on comparable problems. Therefore there exists some benchmark real-world datasets such as UCI[1] or KDD Cup datasets[2] ideal for comparing of performance of algorithms. Besides these real-world datasets there were described synthetic data sets generators which allow to generate statistically similar data sets and which allow to change only some statistic property of generated dataset.

A pattern mining [1] is in general important problem of data mining. The task tries to recognize some patterns in the dataset that occurs more frequently than others. The basic problem of pattern mining is *mining of frequent patterns and association analysis* [2]. Association analysis is abundantly used in market basket analysis to discover habits of customers. Its results can be employed for product recommendation. An example of such a rule can be TV→DVD player telling that when customers buy a TV it's likely that a DVD player will appear in their market basket too.

If the dataset contains an information that reflects that some event occurs before another, than we get sequences. The time extension of mining frequent patterns is a sequential pattern mining [3]. Following an example before, the example of sequence can be ⟨TV DVD⟩ which mean that a customer buys TV and later he buys DVD.

More precisely, the *itemset* e is a non-empty subset of set of all items $\mathcal{I} \in \{i_1, i_2, \ldots, i_N\}$, itemset is called *element* in the context of sequential patterns mining. A *sequence* is defined as an ordered list of itemsets (elements) denoted as $S = \langle e_1 e_2 \ldots e_n \rangle$. For the given sequence database $\mathcal{D}$ and some sequence s, the *support* of sequence s is a percentage of all sequences of $\mathcal{D}$ that contains s. Then the sequence patterns are sequences which has support greater than given threshold called *minimum support*. For formal definitions see [3].

We extended the sequential pattern mining by the concept of hierarchical sequential patterns in [4]. The idea is, that sequential patterns with items of different level has a different support. For example, the sequence ⟨LCD_TV DVD_recorder⟩is not frequent but sequence with generalized items ⟨TV DVD⟩ could be. The generator presented in this paper is designed to prepated testing datasets for this type of data mining algorithms.

The related work is summarized in Section 2. The section is focused on the IBM Quest Synthetic Data Generator. Basic mathematic and statistics tools are defined in the Section 3. Our method of the generator of synthetic dataset is described in Section 4. We verify output datasets and performance of generator in Section 5. Finally, the results are summarized in Section 6.

II. Related Work

A well-known generator of synthetic dataset is IBM Quest Synthetic Data Generator [5]. The last available version supports three types of datasets: association rules, multi-level association rules and sequential patterns. Parameters of each type of dataset is shown in Table I.

Generator details for association rules datasets were described in [2]. Firstly, the sizes of all transactions $|\mathcal{D}|$ are initialized by picking random numbers from a Poisson distribution with mean equal to the generator parameter average transaction size $|T|$. Then, the algorithm generates all potentially large (frequent) itemsets $\mathcal{T}$ by given parameters number of maximal potentially large itemsets $|L|$ and average large itemset size $|I|$ and assign each potentially large itemset

[1]UCI Machine Learning Repository. URL: http://archive.ics.uci.edu/ml
[2]KDD Cup. URL: http://www.sigkdd.org/kddcup/index.php

Table I
PARAMETERS OF IBM QUEST SYNTHETIC DATA GENERATOR FOR ASSOCIATION RULES (AR), SEQUENTIAL PATTERNS (SP) AND MULTI-LEVEL ASSOCIATION RULES (TAX).

Parameter	AR	SP	Tax
Number of Transactions	$\|\mathcal{D}\|$	$\|\mathcal{D}\|$	$\|\mathcal{D}\|$
Avg. size of Transaction	$\|T\|$	$\|T\|$	$\|T\|$
Avg. size of potentially large itemsets	$\|I\|$	$\|I\|$	$\|I\|$
Number of potentially large itemsets	$\|L\|$	N_I	$\|L\|$
Number of items	N	N	N
Avg. length of potentially large seqeucnes		$\|S\|$	
Number of potentially large seqeucnes		N_S	
Number of roots			$\|R\|$
Avg. taxonomy fanout			f
Avg. depth of items in transactions			d

with probability to be generated. Finally, each transaction if filled by selected subset of itemsets from $\mathcal{T}$. Futher, the algorithms reflecting other real-world dataset properties such as corruption of itemsets or inter itemset similarity. However, it is impossible to set some important parameters such as expected threshold of minimum support for large itemsets. Instead of this, the support is inversely related with the $|L|$. In the case of taxonomy generation, the only difference is that items are firstly assigned to taxonomies. The contept of generating asscoation rules datasets was extended to generation of sequence patterns datasets in [6].

Another approach for generating synthetic datasets is presented in [7]. Authors described a method for generating datasets for clustering and outlier analysis combining distribution propertied with transformation functions.

III. PRELIMINARIES

The basic concepts of dataset generators is strongly connected with math, especially statistics [8]. In this section, we describe mathematical tools used for generating of synthetic datasets.

A random variable is a outcome number of a random experiment. *A discrete random variable* is a random variable with a finite or countably finite range. *A continuous random variable* is a random variable with an interval of real numbers for its range.

A. Discrete random variable

For a discrete random variable X with possible values $x_1, x_2, \ldots, x_n$ *a probability mass function* is a function: $f(x_i) \geq 0$, $\sum_{n=1}^{n} f(x_i) = 1$ and $f(x_i) = P(X = x_i)$. The *cumulative distribution function* $F(x)$ of a discrete random variable X is defined as $F(x) = P(X \leq x) = \sum_{x_i \leq x} f(x_i)$. The main *measures* for discrete random variable is mean and variance. The *mean* $E(X)$ of the discrete random variable X is $E(X) = \sum_x xf(x)$. The *variance* $D(X)$ of X is $D(X) = E(X - E(X))^2$.

The distribution of a random variable reflects its measure properties. In the paper, we use a discrete random distribution called Poisson distribution. The random variable X that equals the number of counts in the interval is a Poisson random variable with parameter $0 < \lambda$ which probability mass function $f(x)$ is

$$f(x) = \frac{e^{-\lambda}\lambda^x}{x!} \qquad x = 0, 1, 2, \ldots \tag{1}$$

The *Poisson random variable* X with parameter λ has following mean and variance: $E(X) = \lambda$, $D(X) = \lambda$. We denote a random X variable with Poisson distribution with parameter λ as $Po(\lambda)$. The examples of Poisson random variables with parameters 2 and 5 are shown on Fig. 1. In practice, using of this distribution of random variable guarantees that the number λ will be generated with the highest probability wrt. $E(X)$ and $D(X)$.

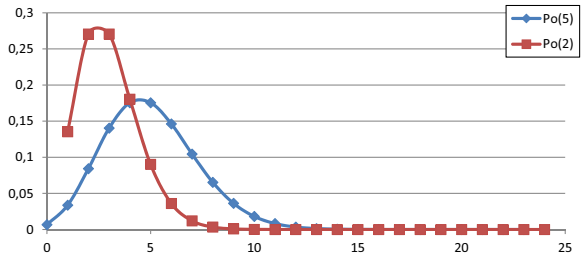

Figure 1. Poisson random variables with different parameters $\lambda = \{2, 5\}$.

B. Continuous random variable

For a continuous random variable X, a probability density function is a function $f(x) \geq 0$, $\int_{-\infty}^{\infty} f(x)dx = 1$ and $P(a \leq X \leq b) = \int_a^b f(x)dx$. A *cumulative distribution function* of a continuous random variable X is defined as $F(X) = P(X \leq x) = \int_{-\infty}^{x} f(u)du$. The *mean* $E(X)$ of the discrete random variable X is $E(X) = \int_{-\infty}^{\infty} xf(x)dx$. The *variance* $D(X)$ of X is $D(X) = \int_{-\infty}^{\infty}(x - E(X))^2 f(x)dx = \int_{-\infty}^{\infty} x^2 f(x)dx - E(X)^2$.

In this part, we will describe next distribution used by our generator. A *continuous uniform random variable* X is a continuous random variable with probability density function

$$f(x) = \frac{1}{b-a} \qquad a \leq x \leq b \tag{2}$$

The continuous uniform random variable has following measures: $E(X) = \frac{(a+b)}{2}$, $D(X) = \frac{(b-a)^2}{12}$. The example of the continuous random variable with parameters $a = 0, b = 1$ is shown on Fig. 2.

C. Taxonomies

Taxonomy structure is a *rooted tree*. A rooted tree $R = (V, E)$ is a tree with one vertex $r \in V$ chosen as its *root*. We

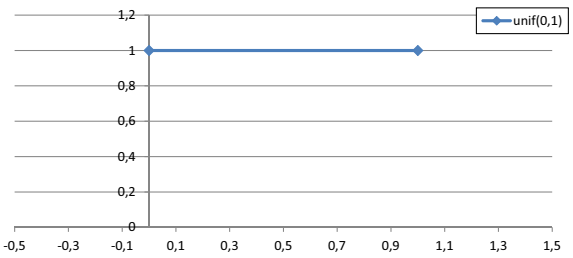

Figure 2. Uniform distribution on interval $(0, 1)$.

refer to a vertex of a rooted tree as a *node*. For each node v in a tree, let $UP(v)$ be the simple unique path from v to r. If $UP(v)$ has exactly k edges then the *level* of v is k for $k \geq 0$. The level of the root is 0. The *height* of a taxonomy is the greatest level in the tree. The *parent* of $v \neq r$, formally $parent(v)$, is neighbour of v on $UP(v)$. The parent of r is not defined. We say if v is the parent of u then u is a *child* of v. A *leaf* is a node having no child [9].

Let τ be a nonempty set of taxonomy structures such that nodes in taxonomy structures represent items and each item $i \in I$ appears in exactly one taxonomy structure $R \in \tau$. Notice that we do not require that items need to be only leaf nodes.

IV. Generator of Hierarchical Sequences

In this section, our algorithm for generating hierarchical sequential datasets is described. We specify the set of settable parameters and the process of generating of the dataset (referred simply as generator).

A. Parameters of the Generator

The list of parameters of our generator is in the Table II. The parameters different to generator presented in [2], [6] are in bold. The main difference is that our generator allows to specify expected (minimum) threshold $Supp_S$ for strong sequences in the output dataset. Note, that the dataset must have transactions of enough average length othervise the support will not be abide (e.g. strong sequences of length $|S|$ cannot have support 60% if the average length of sequences is equal to $|S|$).

Next differences are in the definition of taxonomies. Number of children of each taxonomy node is counted wrt. to the parameters R_d, $|R|$ and N (details are presented in the next subsection). The level of items inside the strong sequences and sequences are managed by parameter P^R_{ch}, respectively P^S_{ch}.

B. The method of the generator

The generator works basically in two phases:

Table II
Parameters of our hierarchical sequence generator.

Parameter	H-SP
Number of transactions (sequences)	$\|\mathcal{D}\|$
Avg. size of transaction (sequence)	$\|T\|$
Avg. size of strong itemsets	$\|I\|$
Number of items	N
Avg. length of strong sequences	$\|S\|$
Number of strong sequences	N_S
Avg. support of strong sequences	$Supp_S$
Number of taxonomies (roots)	$\|R\|$
Avg. taxonomy height	R_h
Probability of children (general items)	P^I_{ch}
Probability of children (items of strong sequences)	P^S_{ch}

1) preparation of the result model – in this phase, all the strong sequences with the probability of occurrence are prepared,
2) generating of the output dataset – this phase generates sequences into the output sequence database.

In following paragraphs we will describe each step of generator in more details.

1. Create Hierarchies. The very first step of the generator is to create $|R|$ hierarchies and the set of all possible items in the dataset. The generator assign to each each taxonomy R a count of items of taxonomy $N_{R_i} = Po(\frac{|N|}{|R|})$. The sum of all items $N^{rand} = \sum_{R_i \in \tau} N_{R_i}$ generated by the random generator is probably not equal to required number of items N, therefore the numbers N_{R_i} are normalized from interval $1, \ldots, N^{rand}$ to interval $1, \ldots, N$. Than, the optimal number of children ch of each node is evaluated expecting the tree R is a full ch-ary tree. Note, that the total count of nodes in the full ch-ary tree of height R_h is $N_{R_i} \approx (ch^{R_h+1} - 1)/(ch - 1)$. Finally starting with a root nodes R, the number of $Po(ch)$ children are generated for each tree node recursively.

2. Generate hierarchical sequential patterns model. The result sequential patterns, which would get a data mining algorithm applied on the dataset, are are prepared in this step. The generator prepares N_S sequences and initializes theirs lengths by numbers get from a random variable with distribution $Po(|S|)$. Then, elements and items of sequences are generated. Firstly, the generator randomly selects the taxonomy R_i. Then the concrete item from R_i is selected randomly by top-down traversing a tree. Next child-node is selected with probability P^I_{ch} during traversing if any child exists, otherwise actual item is added into the last element of sequence. If the number of items in actual element reach the number get by $Po(|I|)$, new element added into the sequence. Finally, each sequential pattern is associated with its probability of selection.

3. Generate the result dataset. The generator generates $|\mathcal{D}|$ dataset sequences. The length of each dataset sequence is set to random number $Po(|S|)$. The dataset sequence can contain items of some sequential pattern or a noise (random items). While dataset sequence is being generating, it is decided, if the next generated item should be noise or next sequential pattern. The probability of generating sequential pattern is related with

the support of sequence – if the support is higher, then the noise probability should be lower. The probability of noise in our generator is $P_{NOISE} = \frac{N_S * Supp_S * |S|}{|T|}$. The $1 - P_{NOISE}$ probability ensures reservation of necessary number of items to reach sufficient support of sequential patterns. The sequential pattern to be generated is selected randomly wrt. probabilities of sequential patterns. If the generator process of generating sequential pattern is started, the items of the sequential pattern are copied into the sequence. If the actual selected item has any child, the item is replaced by its child with probability P^I_{ch} (respectively with P^S_{ch} for items of patterns) recursively. This allows generate various hierarchical sequence datasets. In addition, the items can be interleaved with noise of by the gap probability or the item can be replaced randomly by noise according to the corruption level.

V. Experiments and Verification

VI. Conclusions

In this paper

Acknowledgement

This work has been supported by the research programme TAČR TA01010858, BUT FIT grant FIT-S-11-2, by the research plan MSM 0021630528, and the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

References

[1] J. Han and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2006.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. of 20th Intl. Conf. on VLDB*, 1994, pp. 487–499.

[3] ——, "Mining sequential patterns," in *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, 1995, pp. 3–14.

[4] M. Šebek, M. Hlosta, J. Kupčík, J. Zendulka, and T. Hruška, "Multi-level sequence mining based on gsp," *Acta Electrotechnica et Informatica*, no. 2, pp. 31–38, 2012.

[5] IBM, "Ibm quest synthetic data generator," 2010. [Online]. Available: http://sourceforge.net/projects/ibmquestdatagen/

[6] R. Agrawal and R. Srikant, "Mining sequential patterns," IBM Research Division. Almaden Research Center, Tech. Rep.

[7] Y. Pei and O. Zaïane, "A synthetic data generator for clustering and outlier analysis," Department of computing Science, University of Alberta, Tech. Rep.

[8] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 4th ed. John Wiley & Sons, May 2006.

[9] S.-I. Nakano, "Efficient generation of plane trees," *Inf. Process. Lett.*, vol. 84, no. 3, pp. 167–172, Nov. 2002.