International Journal of Pure and Applied Mathematics
Volume 117 No. 22 2017, 131-135
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
Special Issue

ijpam.eu

Using Sequential Pattern Mining for Building Efficient Massive Open Online Courses

Maxim Dunaev, Konstantin Zaytsev
National Research Nuclear University MEPhI, 115409, Moscow, Russia

Abstract - The abrupt increase in the volume of stored data in recent years makes manual analysis virtually impossible, and makes everybody resort to intelligent computer data analysis. Holding Massive Open Online Courses (MOOC) within the framework of projects Coursera, EdX, Udacity and others made it possible to train over 35 million people in the framework of various courses. In this regard, an opportunity appeared to control the learning process and to build efficient sequences of studied subjects, based on the experience of previous training of millions of students. Such sequences are built with the use of Sequential Pattern Mining (SPM) algorithms. However, due to their abundance, identifying the most efficient one for processing sequences of various structures is not so simple. Based on the proposed tests, the article examines efficiency of SPM algorithms of three types – those of horizontal and vertical formats, and "Frequent Pattern-Growth" for building sequences of studied courses in MOOC projects. In comparing the algorithms, the focus has been placed on their performance, since one has to deal with large volumes of data. As a result of the studies performed for homogeneous source arrays with few transactions, the FP-Growth algorithm proved to be the best for all values of support. For inhomogeneous initial data arrays for almost all additional test conditions PrefixSpan algorithm proved to be the best.

Keywords: Open Education, MOOC, sequences, sequential pattern mining, frequent pattern mining, items mining, searching patterns, consequential events, associate rules, a priory loading, vertical format, horizontal format, pattern-growth.

1 Introduction

In the last 5-7 years, due to the explosive growth of the volume of stored information, more and more attention has been paid to intellectual analysis of stored data in various fields of human activities [1]. The data obtained as a result of analysis may be presented as Sequential Patterns, as properties of data (e.g. decision trees), as clustering (e.g. as Chernoff faces), etc. This article is devoted to representation of knowledge as Sequential Patterns. Searching for Sequential Patterns is one of the fundamental goals of Data Mining [2]. This area of activities starts with the article of Agrawal and Shrikant [3]. Sequential Pattern Mining is widely used today in various fields, for example, in analyzing web clicks of users [4], in bioinformatics [5], in market [6] and text [7] analysis, mining [8] and many other areas of human activity.

Over the last 5 years, in Massive Open Online Courses (MOOC) projects like Coursera, EdX, Udacity etc., over 35 million people in more than three thousand courses have been trained [9-11]. The data obtained in these projects make it possible to analyze the chains of already studied training courses, and to create efficient sequences of studied disciplines for new students. For example, for the following courses (Table 1), such sequences have been met over the past years: AB, CB, D, ACB, F, AD, DFE, etc.

Discipline		*Organizer*
Name	*Designation*	
Basics of cyber security and the creation of secure software networks	A	Maryland University in College Park
Cryptography	B	Stanford University
Exit to the Internet with the help of embedded systems	C	University of California
Algorithms: Design and Analysis, Part 1	D	Stanford University
Programming Paradigms (CS107)	E	Stanford University
Programming Methodology (CS106A)	F	Stanford University

Table 1. Examples of courses of the Coursera project

Study of millions of online courses will allow to determine the formed sequences of the studied courses in various areas of training. This means that a student who chooses a particular set of courses may be advised to study one or several extra courses to obtain comprehensive knowledge in the chosen or related area, thereby reducing the total time of forming the educational trajectory. To detect such sequences, special Sequential Pattern Mining (SPM) algorithms are used, which are dependent on the methods of representing initial data and different in the required computing resources. This article compares such algorithms for various types, and identifies the most efficient one in terms of searching for the most efficient sequences of online training courses.

To ensure reliability, events within the studied course are recorded into the event log of MOOC projects that have various semantic structures. Therefore, in order to test SPM algorithms under the same conditions, it is necessary to either adjust the source code of the existing algorithms, or to develop a special parser application that presents the source data in the format that is acceptable for a particular algorithm.

Another challenge here is availability of a sufficiently representative set of SPM algorithms and active emergence of new, more efficient algorithms, as well as availability of publications where separate, often two algorithms of certain type are compared by previously selected more winning tests for one of them.

For example, there are comparisons of SPAM and PrefixSpan [12], GSP and PrefixSpan, [13], TreeProjection and FP-growth [14] algorithms, and many others [15, 16].

In this article the focus is made on comparison of the best in their types of algorithms SPADE, PrefixSpan, FP-growth in solving the task of building sequences of studied

courses in the Massive Open Online Courses projects in processing sets of events of various semantic structures.

2 Materials and Methods

Let us describe the formal statement of Sequential Pattern Mining problem, and then a pseudo-code of SPM algorithms, which we will compare to each other when working with sets of events of various semantic structures.

2.1 Formulation of the problem

There is a training sample represented as matrix "objects, symptoms". Let X be the object space;
$F = \{f_1,\ldots,f_n\}$ – is the space of binary attributes (items), where $f_j: X \to \{0,1\}$;
$X^l = \{x_1,\ldots,x_l\} \subset X$ is the training set.
Each subset (set of attributes) $\varphi \subseteq F$ corresponds to a conjunction:

$$\varphi(x) = \bigwedge_{f\in\varphi} f(x), x \in X, \quad (1)$$

If $\varphi(x) = 1$, then "attributes from φ are also found in set x".
Let us introduce a concept like frequency of occurrence or support for φ in sample X^l

$$v(\varphi) = \frac{1}{l}\sum_{i=1}^{l} \varphi(x_i), \quad (2)$$

If $(\varphi) \geq \delta$, then "set φ is frequent", where parameter δ is the minimum support (Minsupp).
The association rule $\varphi \to y$ is a pair of disjoint sets $\subseteq F$, such that:

a) Sets φ and y are often found together, i.e. $v(\varphi \cup y) \geq \delta$;

b) If φ is found, y is also often found,

$$v(y|\varphi) \equiv \frac{v(\varphi \cup y)}{v(\varphi)} \geq \aleph, \quad (3)$$

where $v(y|\varphi)$ is the confidence of the rule;
$\aleph$ is the minimum confidence (MinConf).
Example: if dataset i - "φ" is processed, dataset j - "y" will also be processed with probability $\nu(y|\varphi) = 60\%$; both datasets will be processed with probability $\nu(\varphi \cup y) = 2\%$.

From (1), we have the antimonotone property: for any ψ, $\varphi \subseteq F$ $\varphi \subseteq \psi$ it follows that $v(\varphi) \geq v(\psi)$. Consequences:

a) if ψ is frequent, all its subset $\varphi \subset \psi$ are frequent,

b) if φ is not frequent, then all sets $\psi \supset \varphi$ are also not frequent,

c) $v(\varphi \cup \psi) \leq v(\varphi)$ for all φ, ψ.

The task of finding association rules consists of two stages. First, searching for frequently met (further referred to as "frequent") sets, followed by searching for association rules for these sets. The search for frequent sets is based on viewing the entire transaction database, while searching for association rules, it is sufficient to use a simple procedure in RAM (in memory).

We need the pseudo-code of analyzed algorithms to understand in which steps some of them are superior to others, and why.

2.2 Algorithm Apriori

The idea of the algorithm involves the following four steps.
Step 1. Allocating one-element (single) frequent subsequences.
Step 2. From the list of single subsequences, we generate lists of two-element subsequences by joining sequences to one-element sequences while they are listed one by one.
Step 3. From the list of two-element subsequences we generate a list of three-element subsequences by the analogy with step 2.
Step 4. The subsequences satisfying the Minsupp value are chosen.

The pseudo-code of the algorithm can be seen below.

Input: X^l – is the training sample;
minimum support for δ.
Output: R ={(φ,y)} is a list of association rules;
1: the set of all frequent source characteristics:
G_1: ={f ∈ F | ν(f) ≥δ};
2: **for all** j= 2,...,n
3: the set of all frequent sets of power j:
G_j: = {φ ∪ {f} | φ ∈ G_{j-1}, f ∈ G_1, ν(φ ∪ {f})≥δ};
4: **if** G_j= ∅ **then**
5: **exiting** the loop on j;
6: R:=∅;
7: **for all** ψ ∈G_j, j= 2,...,n
8: AssocRules(R,ψ, ∅).

2.3 Algorithm SPADE

The idea of the algorithm consists in building a vertical database (table), where the following will be specified for each one-element sequence:

- sequence identifier (SequenceID or SID);
- element identifier (ElementID or EID) is the sequence number of a subsequence in a large sequence, which can be separated by timestamps (timei) or some separators, for example, brackets;
- a set of elements (items).

Next two-element tables are obtained from one-element tables with a "join" operation.

The pseudo-code of the algorithm can be seen below.

Input: C – is the Atomset;
minimum support for δ;
***Output*:** F – is a list with frequent sets;
SPADE (A, δ, F)
1: **for all** A_i∈C
2: T_i← {}
3: **for all** A_j∈C, j≥I and **all** combinations of B from A_i,A_j
4: L(B) = Temporary TID-list of join of L(A_i) from L(A_j)
5: **if** Supp(B) ≥ δ **then**
6: T_i← T_i ∪ {B}
7: F=F∪B
8: Spade(T_i , δ , F)

2.4. Algorithm PrefixSpan

The idea of PrefixSpan algorithm is to first find all frequent items in the original database and add them to the current template, thereby obtaining new frequent sequences, and then to search for frequent sequences of greater length based on projected databases.

In order to find all patterns of sequential events in database D, PrefixSpan(<>, D) is to be invoked. Creating projected databases may greatly affect performance when working with large amounts of data, therefore, instead of physical creation of projections, the so-called pseudo-projection is used. In case of recursive invocation of the PrefixSpan method, instead of the created projection, it is

passed pointers to the minimum position possible occurrences of the elements into client sequences after the current template. A set consisting of the client ID, the transaction ID in the client sequence and the position in the transaction is considered to be a pointer. Due to the pseudo-projection, the speed of the algorithm operation is significantly increased. In addition, running the algorithm requires much less memory.

The pseudo-code of the algorithm can be seen below.

PrefixSpan(s, D|s):

Input: s – is the pattern of consecutive events;

$D|_s$ – is s-projection of the original database D if s is not an empty sequence <>, otherwise $D|_s = D$.

Output: s – is the pattern of sequential events.

1. Find all frequent items b from $D|_s$, such that:

A) b may be attached to the last substantive set of s, forming a frequent sequence;

B) subject set (b) may be added to s, forming a frequent sequence.

2. For each frequent item b:

A) Add b to s, forming new pattern s';

B)Add s' to the result;

3. For each new template s':

A) Build s'-projection $D|_{s'}$;

B) Invoke PrefixSpan(s', $D|_{s'}$);

2.5 Algorithm FP-growth

The basis of this algorithm is a new data structure in the form of a prefix tree. First, a prefix tree (phase 1) is built, then frequent sets are searched for in it (phase 2).

The pseudo-code of the algorithm is shown below.

Input*:* X^l – is the training sample;

Output*:* FP-tree T, $<f_v, c_v, S_v>_{v\in T}$;

1: normalize symptoms $f\in F : \nu(f) \geq \delta$ in descending order ν(f);

Stage 1: building an FP-tree T on sample X^l

2: **for all** $x_i \in X^l$

3: v:=v0;

4: **for all** $f \in F$ such that $f(x_i) \neq 0$

5: **if** there is no child vertex $u\in S_v: f_u=f$ **then**

6: create new vertex u; $S_v:=S_v \cup \{u\}$; $f_u:=f$; $c_u:= 0$; $S_u:= \emptyset$;

7: $c_u: =c_u+ 1/\ell$; v: =u;

8: Stage 2: recursive search for frequent sets by the FP-tree FP T-find(T, ∅,∅);

Input*:* FP-tree T, set $\varphi \subset F$, list of rules R;

Output*:* add all frequent sets that contain φ to R;

1: **PROCEDURE** FP-find(T,φ,R);

2: **for all** $f \in F : V(T,f)\neq\emptyset$ by levels from **bottom to top**

3: **if** $C(T,f) \geq \delta$ **then**

4: add frequent set $\varphi\cup\{f\}$ to list R:

R:=R ∪ {φ ∧ f};

5: build a conditional FP-tree T':=T|f, namely

T':= FP-tree by sub-sample $\{x_i\in X^l: f(x_i) = 1\}$;

6: from T', find all frequent sets, including φ and f:

FP-find (T′, φ∪{f}, R);

Conditional FP-tree T':=T|f may be built quickly, by using only the FP-tree T and without looking into the sample.

Input*:* FP-tree T, attribute f∈F;

Output*:* conditional FP-tree T'=T|f;

1: leave in the tree only the vertices on the paths from vertices v of attribute f from bottom up to root v_0:

$$T' := \bigcup_{v\in V(T,f)} [v, v_0];$$

2: raise the value of counters c_v from vertices $v \in V(T',f)$ bottom-up according to rule

$$c_u := \sum_{w\in S_u} c_w \text{ for all } u \in T';$$

3: remove from T' all vertices of attribute f; their subtrees are not required and are even not created, since at the moment FP-find invocation, all sets that contain elements below f have already been viewed.

2.6 Associative Rules

On the example of continuing the Apriori algorithm, we will show how association rules are built on found sets. This procedure is versatile and may be applied for all considered algorithms.

Input*:* R – is the list of association rules;

(φ,y) – is the associative rule.

Output*:* R – is the list of association rules;

(φ,y) –is the associative rule.

1:**PROCEDURE** AssocRules (R, φ, y);

2:**for all** f∈φ

3: φ′: =φ\ {f}; y′:=y∪ {f};

4: **if** $\nu(y'|\varphi') \geq$ ℵ **then**

5: add associative rule (φ',y') to list R;

6: **if** |φ'|>1 **then**

7: AssocRules (φ′,y′);

The provided pseudo-code of the algorithms allows revealing the following ideas, which are easily visible in texts:

1) the Apriori algorithm uses almost full enumeration of variants, which will presumably lead to bad performance during the test.

2) the use of a vertical data format by algorithm SPADE will supposedly lead to good performance in processing frequent sequences with many datasets;

3) the PrefixSpan algorithm uses pseudo-projection, which implies good performance when working with non-homogeneous data.

4) the FP-growth algorithm builds prefix trees, which will supposedly result in good enough performance when working with homogeneous data.

3 Conclusion

This work is devoted to identifying areas of efficient use of SPM algorithms for resolving the problem of finding efficient sequences of studied courses in such MOOC projects as Coursera, EdX, Udacity, etc. For this purpose, at the stage of preliminary filtering, the best in their class algorithms were selected, like SPADE, PrefixSpan, FP-growth.

For comparing the algorithms, their pseudo-code was given, and special test samples of the original data sets were prepared, which were characteristic of open online learning in such MOOC projects as Coursera, EdX, Udacity and Open Education (courses of Russian universities), and different in their semantic structure.

Comparison of the algorithms for homogeneous initial data sets with few (10-20) transactions (courses) for all values of support identified the FP-growth algorithm as the leading one. When the number of transactions increases, this algorithm remains the best. The difficulty of using this algorithm arises when value MinSupp is <60%, which is explained by an

increased size of built prefix trees to such a magnitude, where RAM is filled quickly (in our experiments, it was 190 seconds). Therefore, obtaining more accurate results requires using a more powerful computer.

Comparison of algorithms for heterogeneous source data sets for almost all additional testing conditions identified the PrefixSpan algorithm as the best one.

In conclusion, based on the built patterns of sequences of studied courses, association rules for iterative formation of efficient sequences of courses with the use of Coursera, EdX, Udacity and other MOOC have been determined.

Using the results of this work in such MOOC projects as Coursera, EdX, Udacity, etc. will allow controlling consecutive study of courses in MOOC projects by various categories of students, using the accumulated data from previous cases for forming efficient chains of studied courses.

Acknowledgements

This work was supported by the Competitiveness Program of National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), contract with the Ministry of Education and Science of the Russian Federation No. 02.A03.21.0005, 27.08.2013.

References

[1] Frasconi, P., Landwehr, N., Manco, G., Vreeken, J. “Machine Learning and Knowledge Discovery in Databases”. European Conference, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part III.

[2] Aggarwal C.C. “Recommender Systems: the textbook”. Heidelberg: Springer, 2016.

[3] Agrawal R. and Srikant R. “Fast algorithms for mining association rules”. The International Conference on Very Large Databases, pp. 487–499, 1994.

[4] Singh, Harpreet, Manpreet Kaur, and Parminder Kaur. “Web page recommendation system based on partially ordered sequential rules”. Journal of Intelligent & Fuzzy Systems, 32 (4), 3009-3015, 2017.

[5] Krishnan R., Nair A.S., Dhar P.K. “Computational study of ‘HUB’ microRNA in human cardiac diseases”. Bioinformation 13 (1), 17-20, 2017.

[6] Maji, Giridhar, Soumya Sen, and Amitrajit Sarkar. “Share Market Sectoral Indices Movement Forecast with Lagged Correlation and Association Rule Mining”. In IFIP International Conference on Computer Information Systems and Industrial Management, pp. 327-340. Springer, Cham, 2017.

[7] Maylawati, Dian Sa’adillah, and Putri Saptawati. “Set of Frequent Word Item sets as Feature Representation for Text with Indonesian Slang”. In Journal of Physics: Conference Series, 801 (1), 012066. IOP Publishing, 2017.

[8] Bai, Pavitra, and Ravi Kumar G.K. “Efficient Incremental Itemset Tree for approximate Frequent Itemset mining on Data Stream”. In Applied and Theoretical Computing and Communication Technology (iCATccT).