

METHOD FOR CLASSIFYING USER ACTION SEQUENCE

FIELD

The present techniques relate to the methods for detecting fraud, particularly for detecting legal/fraudulent user actions.

DESCRIPTION OF THE RELATED ART

Anti-fraud, or fraud monitoring systems are used to assess financial transactions in the Internet that may be suspected of fraud and offer recommendations on their further processing. Usually, anti-fraud services comprise rules, filters and lists, both standard and unique, and each transaction is checked against them.

Virtually every anti-fraud system operates with specified limits that are based on simple rules, such as:

- limits on the number of purchases using the same bank card of by the same user during a specified period of time;
- limits on the maximum sum of money spent on a single purchase using a single bank card or by the same user during a specified period of time;
- limits on the number of bank cards used by the same user during a specified period of time;
- limits on the number of users using the same bank card;
- purchase histories for each bank card or user (so-called blacklists or whitelists).

This approach fails to determine whether an operation is legal or fraudulent.

Another known solution involves signatures. Said signatures have to be updated regularly in order to keep determining whether an operation is legal or fraudulent at a practical level. Usually, such signatures are generated by a human analyst who analyzes action sequences for fraudulent features and determines their characteristics, which is labour intensive and may depend very much on the analyst's competence as well.

SUMMARY

The present techniques may eliminate the drawbacks of the known solutions.

The present techniques may allow to automatically and precisely determine features associated with specific operation classes.

The method for classifying user action sequence that is performed on a computer device (see Fig. 2), comprises the following steps:

- setting triggers on interface elements and events associated with those elements and other user actions;
- recording a user action sequence with accompanying events, along with the timestamps of said actions or events, and assigning a default class to the sequence;
- obtaining time data, user data, and the actual class the given sequence of user actions at the given time belongs to, and replacing its default class with the actual class;
- generating a training set comprising user action sequences belonging to different classes;
- subdividing the user action sequences in the training set into subsequences, based on the following parameters: max subsequence length, min subsequence length, number of the most informative features (n);
- for each subsequence, determining the frequency of it belonging to each of the classes;
- scoring each subsequence based on the frequency of it belonging to each of the classes and selecting n most informative subsequences pointing at the probability of a given user action sequence belonging to a specified class;
- in each user action sequence in the training set, searching for n most informative subsequences to generate a set of Boolean features, where each feature shows whether its corresponding subsequence is present in the given action sequence; and
- submitting the generated set of Boolean features as a training set for the classification algorithm.

In an exemplary embodiment, the method for classifying user action sequence (see Fig. 3) that is performed on a computer device, comprises the following steps:

- obtaining a user action sequence with accompanying events, along with the timestamps of said actions or events;
- assigning a default class to the action sequence;
- obtaining time data, user data, and the actual class the given sequence of user actions at the given time belongs to, and replacing its default class with the actual class;
- generating a training set comprising user action sequences belonging to different classes;

- subdividing the user action sequences in the training set into subsequences, based on the following parameters: max subsequence length, min subsequence length, number of the most informative features (n);
- for each subsequence, determining the frequency of it belonging to each of the classes;
- scoring each subsequence based on the frequency of it belonging to each of the classes and selecting n most informative subsequences pointing at the probability of a given user action sequence belonging to a specified class;
- in each user action sequence in the training set, searching for n most informative subsequences to generate a set of Boolean features, where each feature shows whether its corresponding subsequence is present in the given action sequence; and
- submitting the generated set of Boolean features as a training set for the classification algorithm.

BRIEF DESCRIPTION OF THE ATTACHED FIGURES

FIG. 1 illustrates an exemplary embodiment of the system according to the present techniques, particularly, the system performing the method of the present techniques.

FIG. 2 illustrates the flowchart of the method according to the first preferred embodiment.

FIG. 3 illustrates the flowchart of the method according to the second preferred embodiment.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

There is a bank web site 105 with a personal account located on a server 110, where user actions have to be analyzed for legality/fraudulence. Users view website pages on their personal devices 120 via an application 130, e.g. a web browser or a mobile banking app.

In some embodiments, personal devices 120 may include, but are not limited to, a smartphone, a tablet, a PDA, a laptop, or a desktop PC.

In order to determine whether a user action is legal or fraudulent, for each website page (or screen in the mobile application), user interface elements and events to be tracked are defined. In order to track interface elements and events, triggers are set for user interactions with said elements and/or for occurrence of said events.

For instance, in a bank web site, said events may include:

- registration page entering;
- registration;
- registration confirmation;
- registration error;
- registration timeout;

- main page;
- transfer to another client of the bank;
- transfer to a client of another bank;
- transfer to an organization <Name>;
- payments to mobile providers;
- money transfer confirmation;
- cheque issue;
- wrong password;
- timeout error;
- setting viewing (specify);
- insurance viewing;
- card viewing;
- account viewing;
- deposit viewing;
- credit viewing;
- exit;
- phone credit recharge.

User actions may include the following:

- selecting an interface element;
- clicking a left/right mouse button;
- browsing through website pages or application screens (e.g. pages of remote banking);
- switching between browser frames or application tabs;
- clicking a button;
- clicking a check button or radio button element;
- editing an input field, or selecting a list item;
- pressing a keyboard key.

When any of the triggers is set off on the device 120, in the application 130, it stores the data that are associated with that user action or event on the server 110. An example of the data entry: <event_MouseClick>, <01/09/2017 17:53>, <29073>, i.e. a mouse click event occurred at 17:53 on 01/09/2017 in the session of the user 29073.

In some embodiments, the trigger stores the data that are associated with the user action or event on the user device 120.

In some embodiments, events and user actions have codes (e.g. symbols or numbers) associated with them. For instance, the <event_MouseClick> event may be stored as a number, or as a sequence of digits and/or letters.

User ID or other identification data are sent from the server 110 or other servers/systems that identify users (indirectly, as well) to the user device 120.

Interacting with a website, users utilize various interface elements and perform various actions that set off triggers, and their action sequences are recorded along with the events accompanying them.

For example, a user 34029 (an exemplary user ID) may have the following action sequence when interacting with a website on their device 120:

```
<login>, <22/09/2017 16:49:39>, <34092 >  
<click_Payment>, <22/09/2017 16:49:43>, <34092 >  
<transfer_Money>, <22/09/2017 16:50:14>, <34092>
```

Such user action sequences are stored on the server 110, or, in some embodiments, on the user device 120, and also have a default class (e.g. legal action class (G)) assigned to them.

The number of classes depends on the task to be solved. In some embodiments, the following classes are discerned: F - fraudulent actions, and G - legal actions; or, e.g. actions of the user under analysis and actions of all other users; or human actions and bot actions; etc.

Some embodiments may have three or more classes. For example, a class may comprise the actions of each user, or it may comprise the actions of a group of users, such as children, adults, the elderly.

If, at some point, it is found that a user was compromised, i.e. an action or a sequence of actions were performed by a malefactor, then the server 110 (or, in some embodiments, the user device 120) will determine the compromised action sequence and assign it to the class of fraudulent actions (F).

The sequences comprising user actions of different classes make up a training set that is then used to automatically detect fraudulent actions.

In order to automatically detect fraudulent actions, the following steps are taken:

User action sequences are subdivided into subsequences, based on the following parameters: max subsequence length (max_subsequence_length), min subsequence length (min_subsequence_length), number of the most informative features (n). At this step, the classes, to which each subsequence belongs, are known.

Please note that the parameters above may be set directly by the administrator of the server 110, e.g. based on their past experience or professional intuition.

In case the method is implemented on the user device 120, the parameters above may be set by developers or they may be obtained from the settings server.

For instance, in an exemplary embodiment of the present techniques, the parameters may be set (particularly, defined) by exhausting every sensible value, wherein the final result is the proof of sensibility. For instance, the administrator may set max and min values for each parameter and then choose the best ones.

An example of subdivision:

- max_subsequence_length = 6, i.e. max length of a subsequence is 6 in this example;
- min_subsequence_length = 3, i.e. min length of a subsequence is 3 in this example;
- n = 400, i.e. the number of features is 400 in this example.

There is a coded action sequence (s) = ABFDSAADOO, wherein each symbol represents a users action or an accompanying event (other attributes, such as timestamps, or user ID, are not taken into account). This sequence is subdivided into subsequences that have a maximum length of max_subsequence_length = 6, which results in the following set: {ABFDSA, BFDSAA, FDSAAA, DSAAAD, SAAADO, AAADOO}.

Each element of this set is then also subdivided in all sorts of subsequences that have a minimum length of not less than min_subsequence_length, starting from the beginning of the coded user action sequence:

ABFDSA --> {ABF, ABFD, ABFDS, ABFDSA}

BFDSAA --> {BFD, BFDS, BFDSA, BFDSAA}

...

AAADOO --> {AAA, AAAD, AAADO, AAADOO}.

For the last subsequence AAADOO, its endings {AADOO, ADOO, DOO} will also be found.

Then, a general set may be selected, particularly, an ordered set (A) that combines every other set ({ABF, ABFD, ABFDS, ABFDSA}, {BFD, BFDS, BFDSA, BFDSAA}, ... {AAA, AAAD, AAADO, AAADOO}, {AADOO, ADOO, DOO}) into one, wherein the general set (A) comprises the elements a(1), a(2), ..., a(U), where U is the power (number of elements) of the general set (A). Please note that each user action sequence will have its own general set (A) .

The, the server 110 (or, in some embodiments, the user device 120) determines the frequency of each element in the set (A) belonging to a specific class (e.g. the legal action class or the fraudulent action class, in case these two classes are used). For instance, the sequence ABFDS may appear 500 times among fraudulent sequences, and just 10 times among legal ones.

Then, the server 110 (or, in some embodiments, the user device 120) scores each element in the set (A) taking into account its frequency of belonging to each of the classes, selecting n (set in advance) most informative elements that point at the probability of the element in the set (user action subsequence) belonging to the given class.

In some embodiments, the elements are scored using Gini index $(1 - \sum_{i=1}^k p_i^2)$, where k is the number of classes (e.g. two classes: legal actions and fraudulent actions), and pi is the share of the [i] class.

In other embodiments, elements are scored using the number of fraudulent sequences.

In yet other embodiments, elements are scored using the number of legal sequences.

Then, the server 110 (or, in some embodiments, the user device 120) searches for n most informative elements that have been selected earlier in the set (A) for each user action sequence in the training set, thus generating a set of boolean features (a set of n -sized vectors), where each vector element (feature) marks that this element (subsequence) is a part of a given action sequence.

A training set contains the following user action sequence (A): ABFFAADOSDOOG. An example of feature generation: there are three subsequences (elements in the set A) that are the most informative ones: DOO, BFDSA, AAD (at $n = 3$). There will be n selected features.

For this action sequence, the server 110 (or, in some embodiments, the user device 120) conducts a search for DOO, BFDSA, and AAD subsequences, thus generating a set (vector) of n (three, in this example) Boolean features $\{1, 0, 1\}$, where 1 means that the given subsequence is present in the sequence, and 0 means not.

In some embodiments, in order to improve efficiency of the claimed method, a unique Boolean function ($f_1, f_2, \dots, f_n$) is generated for each selected informative feature, the function determining whether the given feature is present in the user action sequence or not.

The generated set of feature vectors, together with the vector classes, is then inputted into the classification algorithm (machine learning algorithm) as a training set.

The training set has to contain sets of features that were reliably identified as belonging to different classes, i.e. feature vectors of legal action sequences and feature vectors of fraudulent action sequences.

Machine learning algorithms may include, but are not limited to, the Random Forest algorithm, the Support Vector Machine (SVM), or any other classification algorithm, including classifier ensembles.

After the classifier (classifier ensemble) has been trained, the server 110 (or, in some embodiments, the user device 120) will be able to use this classifier to determine whether user actions are legal or not. In case user actions have been identified as not legal (fraudulent), the server 110 will be able to take corresponding measures, such as to block the user, cancel an operation, etc., whatever is appropriate in a given situation.

As a rule, after the classifier (classifier ensemble) has been trained, the server 110 determines the class the chain belongs to, as follows:

recording the user action sequence; determining the vector of boolean features using n informative features of the sequence in question, selected before; passing the

vector over to the trained classifier, which then returns a priori probabilities of the sequence in question belonging to each of the classes.

Please note that, in an exemplary embodiment, a training set of no less than 2000 fraud facts over the recent days (e.g. 90 days) should be used in order to reliably detect fraudulent actions.

For instance, the training set may be created using data collected over the past several days, e.g. days 85--90. Such data are used to train the classifier, particularly by means of machine learning. Then, the training results are applied to the data from days 1st--5th of the exemplary range in order to test the classifier. Please note that the training may be repeated regularly, e.g. once a week.

Reliability of scoring (particularly, the scoring function) may be increased by creating an ensemble made up of several classifiers, e.g. three classifiers. In this case, for example, the first classifier is trained using the data from days (-90)--(-5); the second classifier is trained using the data from days (-97)--(-13); and the third classifier is trained using the data from days (-104)--(-19). The final scoring or a classifier ensemble may be calculated using the Kolmogorov mean (see https://en.wikipedia.org/wiki/Quasi-arithmetic_mean). For instance, an arithmetic mean can be calculated.

Below is the description of exemplary embodiments of the present techniques.

Please note that the scoring function (used for scoring calculations) is an example of a function for determining informativity. This can be illustrated by the `calc_score` function (shown below) that is written in Python:

```
```python
```

```
def calc_score(g_count, f_count, ratio):
```

```
 """
```

```
 Scoring function
```

```
 :param g_count: legal number
```

```
 :param f_count: fraudulent number
```

```
 :param ratio: the ratio between the total number of legal actions and the total number of fraudulent actions
```

```
 :return: number
```

```
 0 -- equivocal
```

```
 >0 -- likely fraudulent
```

```
 <0 -- likely legal
```

```
 The farther the value is from zero, the likelier its classification is.
```

```

"""
if g_count == 0 and f_count == 0:
 return 0
if g_count == 0:
 return +1.0
if f_count == 0:
 return -1.0
return (float(2*f_count*ratio)/(f_count*ratio+g_count) - 1.0)
"""

```

In an exemplary embodiment of the present techniques, the `calc_score` function operates as follows: if the input of the module is a positive number, then the module returns the same number; if the input of the module is a negative number, then the module returns the number multiplied by (-1).

Also note that the `calc_score` function (or a similar one) may depend on two or more parameters, particularly, on three parameters. In this case, the function may return the informativity number that is directly proportionate to the informativity of a feature. For instance, for three equicardinal classes, the following function (`example3`), written in Python, may be used to calculate informativity:

```

"""python
def example3(a1_count, a2_count, a3_count):
 b1 = a1_count / a2_count if a1_count > a2_count else a2_count / a1_count
 b2 = a2_count / a3_count if a2_count > a3_count else a3_count / a2_count
 b3 = a1_count / a3_count if a1_count > a3_count else a3_count / a1_count
 return min([b1, b2, b3])
"""

```

In an exemplary embodiment of the present techniques, classes are considered equicardinal for a certain set (e.g. a training set), if the set contains equal number of elements (chains) of these classes. For instance, in an equicardinal set with two classes -- fraudulent action class and legal action class -- the number of fraudulent chains is equal to the number of legal chains.

Also, in an exemplary case, for non-equicardinal classes, there may be added at least three values, e.g. `ratio1`, `ratio2`, `ratio3`, which reflect the ratio between the number of elements of each individual class and the number of elements in the largest class. In an exemplary case, one of these three values is always a 1 (one class is the largest,

so its ratio to itself is 1), wherein the following function can be defined, which is a correction to example3 taking into account ratio1, ratio2, and ratio3:

```
```python
def example3_ratio(a1_count, a2_count, a3_count):
return example3(a1_count / ratio1, a2_count/ ratio2, a3_count / ratio3)
```,
```

Using the informativity function (example3\_ratio) n most informative subsequences may be determined (see above).

In an exemplary embodiment of the present techniques, sequence informativity may be determined using another approach. This approach may be illustrated as follows.

A directed forest (gB) is built. Initially, it is empty, i.e. contains no vertices and no trees. Every edge of every tree has two weights corresponding to two classes -- F and G: w(F) and w(G). Each vertex is marked with a letter.

A sequence (s) = ABFDSAAADOO is divided into all sorts of subsequences, wherein each subsequence is (max\_subsequence\_length) long.

For each subsequence:

- traversing all vertices of the graph (gB) in the given subsequence. For instance, for ABFDSAAA there will be a subgraph (gi):  $A \rightarrow B \rightarrow F \rightarrow D \rightarrow S \rightarrow A$ .
- adding a vertex with weights w(F)=0, w(G)=0, if there is none. Then, depending on the class, the corresponding weight is increased by 1. For a fraudulent subsequence (F),  $w(F):=w(F)+1$ ; and for a legal subsequence (G),  $w(G):=w(G)+1$ . Moving on to the next subsequence. After a subsequence is traversed, the algorithm proceeds to the next subsequence.

It results in the graph (gB). Then, a plurality of edges is built, so that the paths to the root are not shorter than min\_subsequence\_length. Each edge is then scored (see above) using the weights w(F) and w(G). Then, n most probable edges (equal to the number of informative features) are selected. Then, for each such edge, a single sequence is built as follows: both vertices or the edge are selected, along with its antecedents. The vertices thus selected are placed in the edge order, where each subsequence refers to an individual feature.

In conclusion, it should be noted that the details given in the description are examples that do not limit the scope of the present invention as defined by the claims. It is clear to a person skilled in the art that there may be other embodiments that are consistent with the spirit and scope of the present invention.

## CLAIMS

What is claimed is:

1. A method to generate a classification training set to train a classification algorithm to determine legality of user actions, by classifying user action sequence that is performed on a computer device, comprising the following steps:

- setting triggers on interface elements and events associated with those elements and other user actions;
- recording a user action sequence with accompanying events, along with the timestamps of said actions or events, and assigning a default class to the sequence;
- obtaining time data, user data, and the actual class the given sequence of user actions at the given time belongs to, and replacing its default class with the actual class;
- generating a training set comprising user action sequences belonging to different classes;
- subdividing the user action sequences in the training set into subsequences, based on the following parameters: max subsequence length, min subsequence length, number of the most informative features (n);
- for each subsequence, determining the frequency of it belonging to each of the classes;
- scoring each subsequence based on the frequency of it belonging to each of the classes and selecting n most informative subsequences pointing at the probability of a given user action sequence belonging to a specified class;
- in each user action sequence in the training set, searching for n most informative subsequences to generate a set of Boolean features, where each feature shows whether its corresponding subsequence is present in the given action sequence; and
- submitting the generated set of Boolean features as the classification training set for the classification algorithm.

2. A method to generate a classification training set to train a classification algorithm to determine legality of user actions, by classifying user action sequence that is performed on a computer device, comprising the following steps:

- obtaining a user action sequence with accompanying events, along with the timestamps of said actions or events;
- assigning a default class to the action sequence;
- obtaining time data, user data, and the actual class the given sequence of user actions at the given time belongs to, and replacing its default class with the actual class;
- generating a training set comprising user action sequences belonging to different classes;

- subdividing the user action sequences in the training set into subsequences, based on the following parameters: max subsequence length, min subsequence length, number of the most informative features (n);
- for each subsequence, determining the frequency of it belonging to each of the classes;
- scoring each subsequence based on the frequency of it belonging to each of the classes and selecting n most informative subsequences pointing at the probability of a given user action sequence belonging to a specified class;
- in each user action sequence in the training set, searching for n most informative subsequences to generate a set of Boolean features, where each feature shows whether its corresponding subsequence is present in the given action sequence; and
- submitting the generated set of Boolean features as the classification training set for the classification algorithm.

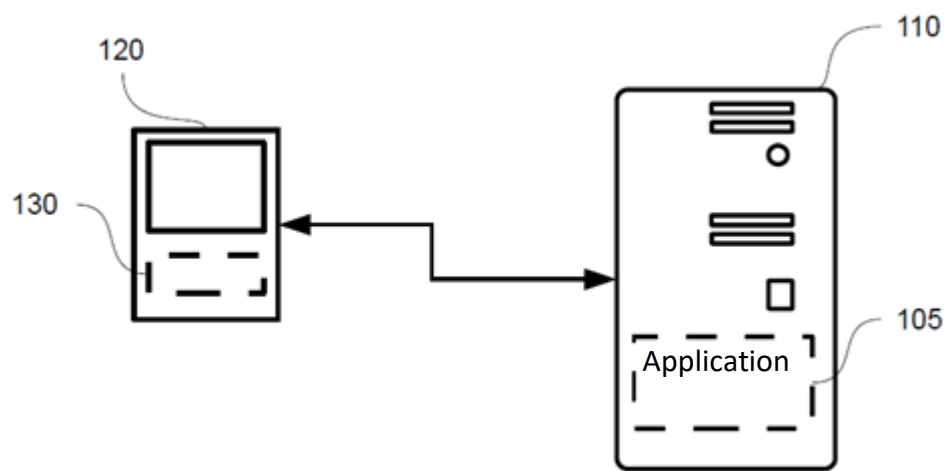


FIG. 1

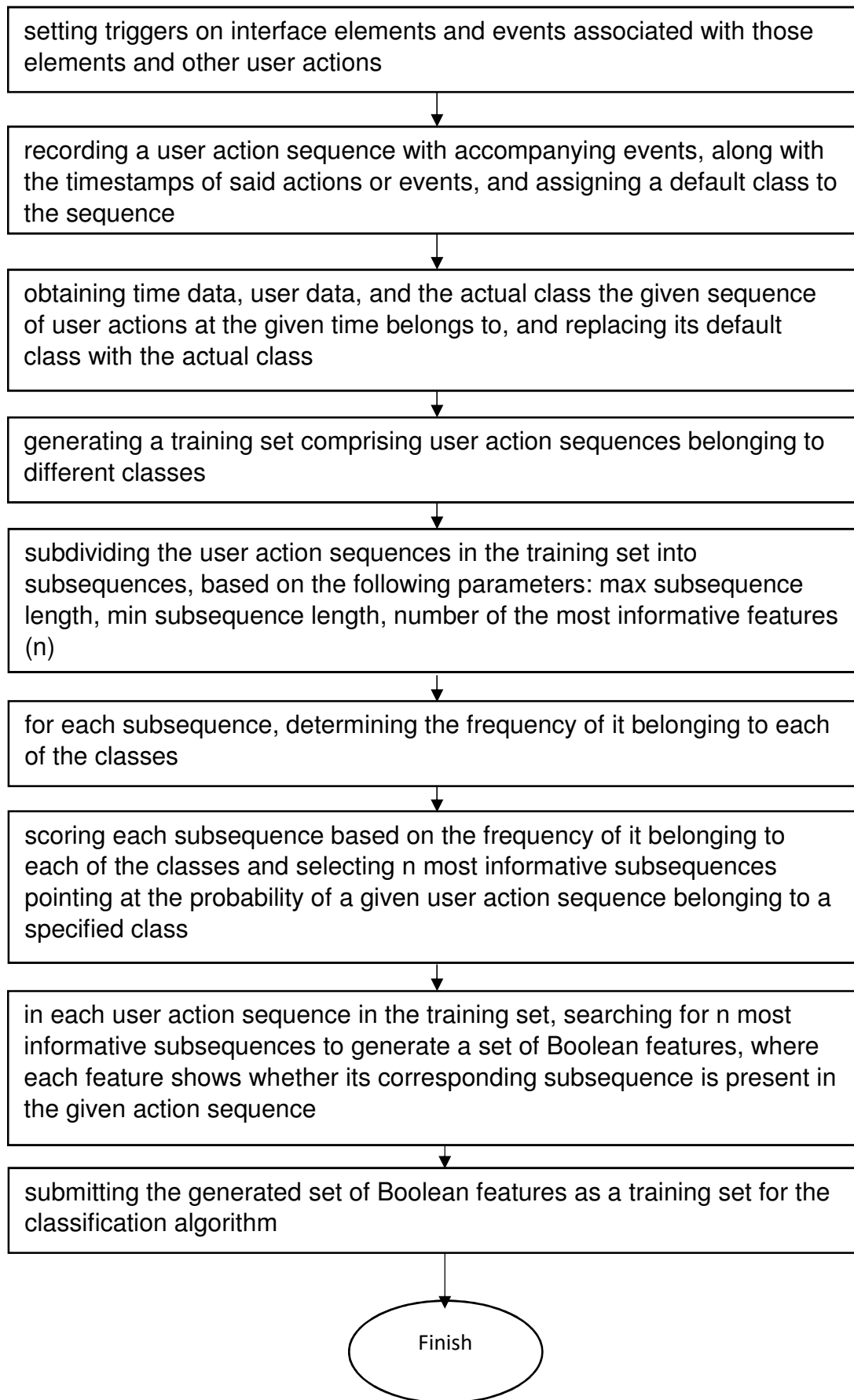


FIG. 2

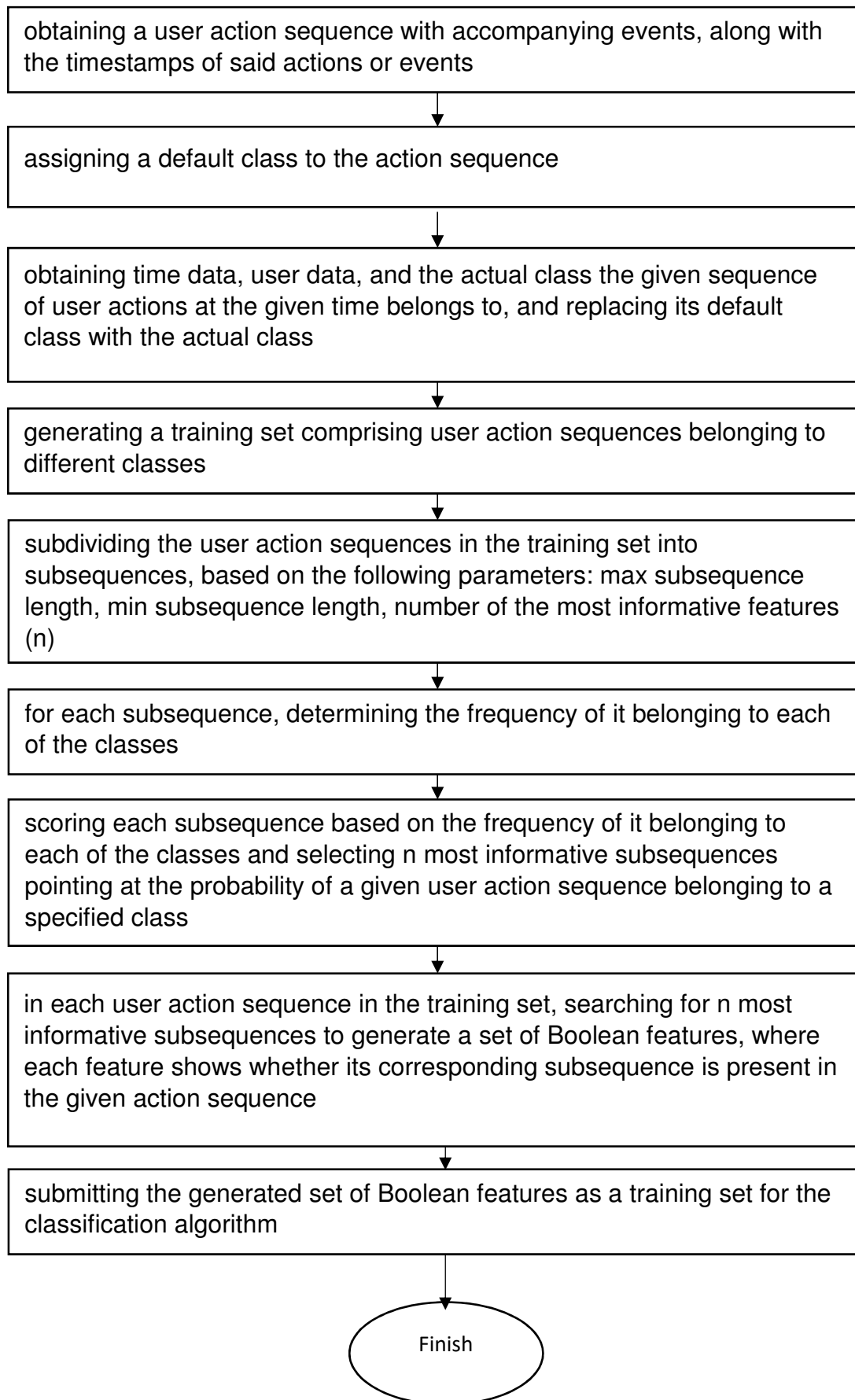


FIG. 3