Citation for published version (APA):

DOI:
10.1109/ACCESS.2019.2925105

Document status and date:
Published: 26/06/2019

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Process Mining in Social Media: Applying Object-Centric Behavioral Constraint Models

GUANGMING LI1,2 AND RENATA MEDEIROS DE CARVALHO1
1Department of Mathematics and Computer Science, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands
2Science and Technology Laboratory on Information Systems Engineering, National University of Defense Technology, Changsha 410073, China

Corresponding author Guangming Li (g.li.3@tue.nl)

This work was supported in part by the National Natural Science Foundation of China under Grant 6190050169.

ABSTRACT The pervasive use of social media (e.g., Facebook, Stack Exchange, and Wikipedia) is providing unprecedented amounts of social data. Data mining techniques have been widely used to extract knowledge from such data, e.g., community detection and sentiment analysis. However, there is still much space to explore in terms of the event data (i.e., events with timestamps), such as posting a question, commenting on a tweet, and editing a Wikipedia article. These events reflect users’ behavior patterns and operational processes in the media sites. Classical process mining techniques support to discover insights from event data generated by structured business processes. However, they fail to deal with the social media data which are from more flexible “media” processes and contain one-to-many and many-to-many relations. This paper employs a novel type of process mining techniques (based on object-centric behavioral constraint models) to derive insights from the event data in social media. Based on real-life data, process models are mined to describe users’ behavior patterns. Conformance and performance are analyzed to detect the deviations and bottlenecks in the question and answer process in the Stack Exchange website.

INDEX TERMS Social media, social event data, process mining, object-centric behavioral constraints, behavioral patterns, OCBC models.

I. INTRODUCTION
Social media is defined as a group of Internet-based applications that build on the ideological and technological foundations of Web 2.0. It allows people to create, share, and/or exchange information and ideas [10]. Some well-known platform examples are Wikipedia, YouTube, Facebook and Twitter. We identify eight different types of social media as follows [3], [9], [10]:

• collaborative projects, e.g., Wikipedia, Scholarpedia,
• blogs and microblogs, e.g., Blogger, Twitter,
• social news, e.g., Digg, Leakernet, Toutiao,
• content communities, e.g., YouTube, Flickr, Instagram,
• social networking sites, e.g., Facebook, LinkedIn,
• opinion, reviews and ratings, e.g., Yelp, Epinions,
• answers, e.g., StackExchange, Quora, and
• virtual worlds, e.g., World of Warcraft, Second Life.

The rise of social media has generated unprecedented amounts of social data. Figure 1 shows how much data are created in one minute in different social media platforms. For instance, Facebook, which is the most active of social networks with over 1.4 billion monthly active users, creates the most amount of social data: users like over 4.1 million posts every minute. Instagram, with 300 million monthly

![Figure 1. The size of data created in one minute in social media.](https://wersm.com/how-much-data-is-generated-every-minute-on-social-media/)

This work is licensed under a Creative Commons Attribution 3.0 License. For more information, see [http://creativecommons.org/licenses/by/3.0/](http://creativecommons.org/licenses/by/3.0/)
users in 2015, comes in second with more than 1.7 million likes on photos each minute.

Due to the large amount of data, social media has become an important source of information for understanding human behavior. It is critical for users, consumers, and service providers to mine social media to extract actionable patterns. Data mining techniques have been widely used in social media to extract insights to improve business intelligence, provide better services and develop innovative opportunities. Representative areas contain community detection [21], information diffusion [29], topic detection and monitoring [30] and sentiment analysis and opinion mining [19].

Although data mining enables people to understand data from various aspects, there is still much space to explore in terms of the event data. In this paper, events refer to users’ behavior in social media platforms and include a broad range of actions: joining a group, becoming friends with a person, posting a photo, sharing a link, updating a status, posting a question, commenting on a tweet, editing a wikipedia article, etc. Events play a significant role in any dynamic system and all these events have a timestamp associated with them. They can be exploited using process mining techniques to derive process-related insights to reflect users’ behavior patterns and operational processes in social media platforms.

Figure 2 shows the framework of applying process mining to social media. The data from social media may be in different forms such as CSV files, database tables or XML files. In the approach used in this paper, they are first parsed and imported into relational databases (after scoping the data based on a particular need) and then transformed into an XOC log that can better organize event data without a case notion [12]. Since classical techniques fail to deal with the social media data which are from more flexible “media” processes and contain one-to-many and many-to-many relations, this paper employs a novel type of process mining techniques (based on Object-Centric Behavioral Constraint Models) to analyze the data [26].

Compared with data mining techniques, our approaches can discover more complex users’ behavior patterns, involving multiple activities, classes and interactions between them, rather than simple association rules. Besides, conformance checking can be applied to detect deviations or undesired behavior, e.g., a question without any answers in Stack Exchange. Moreover, performance can be analyzed to derive insights on the time perspective, e.g., most answers are given in one day after the corresponding questions are posted. Besides the described discovery and conformance checking techniques, we show how the performance analysis approach can be useful.

The remainder is organized as follows. Section II explains the motivation for applying process mining to social media data. In Section III, we introduce the social media data, import them into database tables and transform tables into XOC logs. Section IV explains OCBC models by describing the process of making friends in Facebook. Section V illustrates the discovery approach by discovering an OCBC model from the XOC log explained in Section III. Sections VI and VII analyze conformance and performance in the context of Stack Exchange, respectively. Section IX reviews the related work while Section X concludes the paper.

II. MOTIVATIONS FOR PROCESS MINING IN SOCIAL MEDIA

Process mining has emerged in recent year, which bridges the gap between Business Intelligence (including data mining and machine learning) and Business Process Management. It is more powerful to extract process-related insights from event data, e.g., discovering process models, identifying bottlenecks and detecting violations [25].

Traditional process mining focuses more on the case-centric data generated by structured business processes or systems, in which events are organized as process instances with a clear case notion. However, process mining should not be limited to these data due to its essentially general applicability. Process mining could also be applied to analyze events from various sources as in the Internet of Events (IoE), term proposed in [24]. Typically, these sources are composed of:

- the Internet of Content (IoC), e.g., traditional web pages, articles, encyclopedia like Wikipedia, YouTube, e-books and newsfeeds;
- the Internet of People, e.g., interactions of people, such as e-mail, Facebook, Twitter, forums, LinkedIn;
FIGURE 3. The objects and recorded events in Stack Exchange.

- the Internet of Things (IoT), e.g., things may have an internet connection or tagged using Radio-Frequency Identification (RFID), Near Field Communication (NFC);
- the Internet of Locations (IoL), e.g., data that have a spatial dimension (geospatial attributes).

Figure 3 shows the interface of the Stack Exchange website. It consists of different objects, such as questions, answers, votes and comments. These objects can be created or updated by users’ operations, which are recorded as events in the corresponding data source. For instance, the red part corresponds to an event in which a question was posted at 9:04 on Jan 3. The orange square indicates an event in which a comment was made for the question at 16:05 on Jan 5. The blue square refers to an event in which an answer was posted to the question at 9:40 on Jan 3. The green square shows an event in which a comment was made for the answer at 15:50 on Jan 3. Besides, there are events such as voting for questions/answers, registering users in the website, getting badges.

Figure 3 only presents a small segment of events available in social media. These events can provide insights into social networks and users’ behavior that were not previously possible in both scale and extent. In general, the research topics in social media can be categorized as follows: (i) Who of Social Media (ii) How of Social Media (iii) When of Social Media, and (iv) What of Social Media. The objective of this paper is to employ process mining to answer questions related to “When of Social Media”.

In addition to its advantages of extracting process-related insights, process mining has potential to deal with social media data in terms of structure and scale. As social media data are generated by humans’ behavior, they are often dynamic and unstructured. Process mining has declarative languages to describe behavior patterns or processes in a flexible manner [1]. For the large data scale, process mining can improve its performance by using decomposition and parallel approaches [5], [23].

III. DATA PRE-PROCESSING

A. SOCIAL MEDIA DATA

The first step to mine social media is to derive data from different social media platforms. The data can be in forms such as CSV files, database tables and XML files. In this section, we mainly discuss the available data from Stack Exchange. These data record the objects and events happened in interfaces like Figure 3.

Stack Exchange publishes its data regularly, named Stack Exchange Data Dump, which enables the public availability of most historical data of Stack Exchange [6].
TABLE 1. Common social media data sources.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack Exchange</td>
<td>archive.org/details/stackexchange</td>
<td>Posts, Users, Votes, Comments, PostHistory and PostLinks</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>dumps.wikimedia.org</td>
<td>Revision and page meta information, no actual text</td>
</tr>
<tr>
<td>MovieLens</td>
<td>grouplens.org/datasets/movielens</td>
<td>Sample of movie ratings</td>
</tr>
<tr>
<td>Amazon Food Reviews</td>
<td>snap.stanford.edu/data/web-FineFoods.html</td>
<td>Historical reviews on Amazon for &quot;Fine Foods&quot;</td>
</tr>
<tr>
<td>Twitter</td>
<td>developer.twitter.com/en/docs.html</td>
<td>Tweets created</td>
</tr>
<tr>
<td>Facebook</td>
<td>developers.facebook.com/docs/graph-api/overview</td>
<td>Posts, events, friends and comments</td>
</tr>
</tbody>
</table>

TABLE 2. Statistics of a real-life data set from Stack Exchange released on September 5, 2018 [6].

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>#Instances</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Badges.xml</td>
<td>badges assigned to users</td>
<td>14,515</td>
<td>1,592 KB</td>
</tr>
<tr>
<td>Comments.xml</td>
<td>comments for posts</td>
<td>7,102</td>
<td>2,508 KB</td>
</tr>
<tr>
<td>PostHistory.xml</td>
<td>histories of posts</td>
<td>17,246</td>
<td>18,095 KB</td>
</tr>
<tr>
<td>Posts.xml</td>
<td>links between posts</td>
<td>302</td>
<td>33 KB</td>
</tr>
<tr>
<td>Posts.xml</td>
<td>posted questions and answers</td>
<td>5,760</td>
<td>9,117 KB</td>
</tr>
<tr>
<td>Tags.xml</td>
<td>tags attached on the questions</td>
<td>287</td>
<td>24 KB</td>
</tr>
<tr>
<td>Users.xml</td>
<td>users registered in the website</td>
<td>15,541</td>
<td>6,770 KB</td>
</tr>
<tr>
<td>Votes.xml</td>
<td>votes for posts</td>
<td>22,338</td>
<td>1,979 KB</td>
</tr>
</tbody>
</table>

It is an anonymized dump of all user-contributed content on the Stack Exchange network and formatted as a separate archive consisting of XML files: “Badges.xml”, “Comments.xml”, “PostHistory.xml”, “PostLinks.xml”, “Posts.xml”, “Tags.xml”, “Users.xml” and “Votes.xml”, as shown in Table 2.

In summary, Table 2 provides a complete trace of all the actions related to the “Artificial Intelligence” topic in Stack Exchange for almost two years, i.e., from “2016-08-02” (when the website was founded) to “2018-09-02”. The “#Instances” column indicates the number of instances for each file. For example, there are 15,541 users in the “Users.xml” file.

Along with Stack Exchange data, other common social media platforms make data available, which is summarized in Table 1. One example of social media platform that also publishes data regularly is Wikipedia. A second example is Twitter, which provides an API for third-party applications to get information about timelines and various other objects. Facebook is a third example, and “Graph API” can be used to get data from this platform. Besides that, it is possible to download the “Activity log” from a particular user to derive all actions performed by him/her, as shown in Figure 14.

B. IMPORTING DATA INTO DATABASES

The social media data derived from different platforms in Section III-A can be transformed into event logs for process mining techniques to discover insights. Before that as we use the OCBC approach in [12], we need to parse and import them into relational databases. Here, we use the data from Stack Exchange (i.e., the XML files in Table 2) as an example to walk through the process.

The goal for mining social media influences on how to parse the raw data. For the Stack Exchange platform, we will consider that the goal is to explore the relations between questions and answers. However, there are no explicit questions and answers in the raw data, as both of them are considered as posts stored in the same file “Posts.xml”. In order to get event logs with objects and events related to questions and answers, the posts need to be separated into two categories: questions and answers. Accordingly, the comments (in “Comments.xml”) and votes (in “Votes.xml”) are also separated in the same way.

Figure 4 presents a segment of the resulting tables after parsing the XML files. Based on common knowledge, the dependency relations between tables are identified. For instance, the foreign key (“q_id”) of “answer” table references the primary key (“id”) of “question” table (indicated by r6). Accordingly, the first record in “answer” table references the first record in “question” table.

Note that, there exist one-to-many and many-to-many relations between different tables. Consider for example the relation between users and questions. A user may post multiple questions, e.g., user 8 posted questions 1 and 1537. A question can also have multiple involved users. For instance, question 1 was posted by user 8 and then user 7488 added a tag on the question, indicated by the first and third records in “history” table. In other words, question 1 corresponds to both users 8 and 7488. In summary, there is a many-to-many relation between users and questions.

C. EVENT LOGS

Based on the tables as shown in Figure 4, events can be derived by assuming that each table with a timestamp column
corresponds to an activity and the values in this column correspond to events of this activity. For instance, “question” table can be considered as the activity “post question” and each value in “creation_date” would be an event of this activity.

Process mining takes event logs as input to derive insights. Event logs are used to record and organize past events. The current standard, i.e., XES log format [28], often assumes a case notion to correlate events. It has problems to organize the events from social media, since it is difficult to identify a case notion due to one-to-many and many-to-many relations in social media. Therefore, we use a novel log format, named eXtensible Object-Centric (XOC) in this section.

Table 3 gives an example XOC log containing 20 events, which is originated from the 9 tables in Figure 4. The approach proposed in [12], which is realized as a plugin in ProM, was used for extracting this XOC log. Each event

### Table 3. An XOC log example as the input for discovery.

<table>
<thead>
<tr>
<th>Index</th>
<th>Event</th>
<th>Activity</th>
<th>Timestamp</th>
<th>References</th>
<th>Objects</th>
<th>Object Model</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>re1</td>
<td>register</td>
<td>2016-08-02 15:38:31</td>
<td>u4</td>
<td>u4</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>2</td>
<td>re2</td>
<td>register</td>
<td>2016-08-02 15:38:36</td>
<td>u4</td>
<td>u4</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>3</td>
<td>pq1</td>
<td>post question</td>
<td>2016-08-02 15:39:14</td>
<td>q1</td>
<td>u4, u8, q1, l1, a3, b7</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>4</td>
<td>pa1</td>
<td>post answer</td>
<td>2016-08-02 15:40:24</td>
<td>a3</td>
<td>u4, u8, q1, l1, a3, b7</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>5</td>
<td>gb1</td>
<td>get badge</td>
<td>2016-08-02 15:56:06</td>
<td>b12</td>
<td>u4, u8, q1, l1, a3, b7, b12</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>6</td>
<td>re3</td>
<td>register</td>
<td>2016-08-02 20:49:54</td>
<td>u357</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>7</td>
<td>gb2</td>
<td>get badge</td>
<td>2016-08-02 20:55:18</td>
<td>b352</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>8</td>
<td>qv1</td>
<td>vote for question</td>
<td>2016-08-02 23:59:59</td>
<td>qv1</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>9</td>
<td>pa2</td>
<td>post answer</td>
<td>2016-08-03 14:39:02</td>
<td>a222</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>10</td>
<td>va1</td>
<td>vote for answer</td>
<td>2016-08-03 23:59:59</td>
<td>av357</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>11</td>
<td>pq2</td>
<td>post question</td>
<td>2016-08-11 09:02:52</td>
<td>q1537</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>12</td>
<td>pa3</td>
<td>post answer</td>
<td>2016-08-11 20:39:12</td>
<td>a1569</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>13</td>
<td>pq3</td>
<td>post question</td>
<td>2016-08-17 11:57:42</td>
<td>q1662</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>14</td>
<td>qv2</td>
<td>vote for question</td>
<td>2016-08-17 23:59:59</td>
<td>qv3574</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>15</td>
<td>mcq1</td>
<td>make question comment</td>
<td>2016-08-22 23:32:32</td>
<td>qc1670</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>16</td>
<td>va2</td>
<td>vote for answer</td>
<td>2016-08-22 23:59:59</td>
<td>av3739</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>17</td>
<td>mcq2</td>
<td>make question comment</td>
<td>2016-09-18 23:43:33</td>
<td>qc2109</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>18</td>
<td>re4</td>
<td>register</td>
<td>2017-05-28 13:41:19</td>
<td>u7488</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>19</td>
<td>gb3</td>
<td>get badge</td>
<td>2017-05-28 13:56:45</td>
<td>b6827</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
<tr>
<td>20</td>
<td>mac1</td>
<td>make answer comment</td>
<td>2017-06-07 07:13:55</td>
<td>ac1744</td>
<td>u4, u8, q1, l1, a3, b7, b12, a157</td>
<td>(r2,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
<td>(r3,u4,q1), (r3,u4,h1), (r5,u4,l1)</td>
</tr>
</tbody>
</table>
corresponds to an activity (in “Activity” column) and has a timestamp (in “Timestamp” column) at which the event took place. Moreover, events are atomic and ordered (indicated by the “Index” column). For instance, event re1 corresponds to the first occurrence of activity register at 15:38:21 on August 2, 2016. Each event corresponds to an object model (in the “Object Model” column) which represents the state of the process just after the execution of the event. Each object model consists of objects (in the “Objects” column) and object relations (in the “Relations” column). For instance, the corresponding object model of pq1 consists of four objects u4, u8, q1 and h1 and three object relations (r2, u8, q1), (r3, u8, h1) and (r5, q1, h1). An object relation is represented by a tuple containing three elements, in which the first one is a relation type, the second one is called the target object and the third one is called the source object. The object relation indicates that a record corresponding to the source object references another record corresponding to the target object through the relation type. In order to relate the behavioral perspective and the data perspective (i.e., events and objects), each event refers to at least one object (in the “References” column). For instance, the first event re1 refers to the object u4.

Figure 5 shows the “graphical” representation for the first five events in the XOC log in Table 3. Each black dot corresponds to an event and the cylinder represents the object model after the event. A grey dot means an object, which corresponds to a record in database tables. An object has a class, and the objects from the same table have the same class, grouped by the round-corner squares. The class is implicitly indicated in Table 3. For instance, object u4 indicates that its corresponding class is user and object q1 indicates that its corresponding class is question. The edges between objects indicate the relations between records. The dotted lines link events and the object referred to by the events. Figure 5 reveals the evolution process of a database corresponding to system, along with the events operated on the system.

IV. OBJECT-CENTRIC BEHAVIORAL CONSTRAINTS MODELS

Traditional media such as newspaper, radio, and television provides almost entirely one-way communication, originating from the media providers to media consumers. In contrast, social media changes the scene from one-way communication to where almost anyone can publish written, audio, or video content to other people [3]. This many-to-many media environment results in many-to-many relations in social media data. For instance, a user in Facebook can participate in multiple groups and each group can involve multiple users.

Besides, social media data have different types of objects (e.g., user, tweet, photo, link, article) and interactions. Hence, meta models (e.g., using UML class models) can be used to describe the structure of objects. Moreover, the involved processes in social media are more flexible and there are no strict or structured business processes to limit users’ behavior—there are only declarative constraints or rules between activities. For instance, a question can be answered anytime after the question is posted, corresponding to a constraint that “post answer” activity should be preceded by “post question” activity.

Due to the complex object types, interactions and flexible behaviors, most traditional process mining techniques fail to deal with social media data. Therefore, object-centric behavioral constraint (OCBC) models [26] and corresponding OCBC techniques [11] were proposed as a novel type of process mining techniques.

An OCBC model combines data models with a declarative behavioral perspective. Data models can easily deal with one-to-many and many-to-many relationships. This is exploited to model complex interactions between different types of instances. Classical multiple-instance problems are circumvented by using the data model for event correlation. The declarative nature describes behavioral constraints over activities like cardinality constraints in data models. The resulting OCBC model is able to describe flexible processes involving interacting instances and complex data dependencies in social media data.

Figure 6 presents an OCBC model which describes the process of making friends in Facebook.
The bottom part corresponds to the data perspective (i.e., class model), consisting of four classes (or entities) in the process: “user”, “request”, “friendship” and “message”. The “user” class corresponds to the people who register in Facebook, the “request” class means friend requests sent by users, the “friendship” class indicates the built relations between users and the “message” class denotes the one-to-one messages between friends.

The relations with cardinalities \( r_1, r_2 \) and \( r_3 \) specify the constraints between classes. More precisely, the square symbol (□) indicates that the constraint should be “always” satisfied while the diamond symbol (◇) indicates the constraint should be “eventually” satisfied. The relations are:

- \( r_1 \), which means that each user eventually sends at least one friend request (it is reasonable as the goal of Facebook is to build friendships between people and share things) and each request always refers to precisely two users (the sender and the receiver);
- \( r_2 \), which shows that each request always refers to at most one friendship and each friendship always refers to precisely one request; and
- \( r_3 \), which indicates that each friendship may correspond to any number of messages and each message corresponds to precisely one friendship.

The top part in Figure 6 describes the behavioral perspective (i.e., activity model) of the process by defining five activities (“register”, “send request”, “reject request”, “make friend” and “talk”) and six behavioral constraints \((c_1, c_2, \ldots, c_6)\). Each constraint has a reference activity (on the side with the black dot), target activity (on the other side) and a constraint type (indicated by the shape of the arrow).

For instance, the reference/target activity of \( c_1 \) is “send request”/“register”. The single arrow towards the dot indicates that the constraint type is “unary-precedence”, which means that each “send_request” event should be preceded by precisely one corresponding “register” event. In contrast, \( c_2 \) specifies a “response” constraint (indicated by the double arrow leaving the dot) between “register” (reference activity) and “send request” (target activity). It requires that each “register” event should be followed by one or more “send request” events. \( c_5 \) represents a “non-response” constraint (indicated by the double arrow leaving the dot with an “X” mark) between “reject request” (reference activity) and “make friend” (target activity). It indicates that each “reject request” event should never be followed by any “make friend” events.

The interactions \((aoc_1, aoc_2, \ldots, aoc_5)\) between data and behavioral perspectives, named AOC relationships, present how activities relate to classes based on the cardinalities on the relationships. For instance, the cardinality □0..1 on \( aoc_3 \) means that each “request” object always corresponds to zero or one “reject request” event and the cardinality 1 means that each “reject request” event corresponds to precisely one “request” object. The other four relationships indicate one-to-one correspondence between events and objects.

V. PROCESS DISCOVERY

Section IV introduced OCBC models. Note that, such models can be discovered from real-life data using a plugin in ProM corresponding to the approach in [11]. Figure 7 shows the discovered OCBC model from the event log in Table 3.

![OCBC model discovered from the log in Table 3, which describes the question & answer process in the Stack Exchange website.](image)

In the discovered OCBC model, the data and behavioral perspectives are described (bottom and top, respectively), as well as the interplay between them (middle). It clearly reveals the involved classes (e.g., “question”), activities (e.g., “post_question”) and constraints in the Question & Answer process in the Stack Exchange website. In this section we briefly explain the process of discovering the OCBC model.

A. DISCOVERY OF CLASS MODELS

The class model is discovered based on the object models in the input log. The bottom of Figure 7 shows the discovered class model, which contains nine classes \( (user, question, \ldots, question) \) and ten class relations \( (r_1, r_2, \ldots, r_{10}) \). Next, we introduce the process of discovering class models.

The classes can be learned by incorporating classes of all objects in the object models of all events. For instance, \( user \) is a discovered class since object models contain objects of class \( user \), e.g., \( u_4 \). The class relationships can be learned by observing object relations in object models of each event. \( r_2 \) is discovered between \( user \) and \( question \) classes since there exist object relations involving \( r_2 \) between \( user \) and \( question \) objects, e.g., \( (r_2, u_8, q_1) \).

For each class relationship, its “always” (“eventually”) cardinalities can be derived by integrating the number of related objects of each reference object in the object model of
each (the last) event. For instance, the discovered “always” cardinality on the q_comment side of r4 is □0..2 (including numbers 0, 1 and 2), since (i) in the object model of event pq1, the question object q1 has no related q_comment objects, (ii) in the object model of event mqc1, q1 has one related q_comment object (qc1670) and (iii) in the object model of event mqc2, q1 has two related q_comment objects (qc1670 and qc2109), and remains two related objects in the latter object models. The discovered “eventually” cardinality on the q_comment side of r4 is □0..2 (including numbers 0 and 2) since in the object model of the last event (i.e., mac1), q1 has two related q_comment objects (qc1670 and qc2109) while q1537 and q1662 have no related q_comment objects. The “eventually” cardinality can be omitted on the graph for simplicity, when it is indicated by its corresponding “always” cardinality.

Note that the cardinalities discovered in Figure 7 are based on the log in Table 3, which is a small part of the whole event log derived from the files in Table 2. As a result, the discovered cardinalities are overfitting. For instance, □0..2 of r6 only aggregates the numbers observed in the small log. Taking the whole log as input, the discovered cardinality on r6 is □0..13, which contains more observed numbers. Note that, even the whole log is still a part of all possible real-life transactions. In order to solve the overfitting problem, we can generalize the discovered cardinalities, e.g., extending □0..13 to □0..13.

The discovered cardinalities present the summarized cardinality information. It is possible to see the details of a discovered cardinality (which is supported by our plugin). The panel in Figure 8 presents the distribution of the discovered cardinalities on class relationships. There are four drop-down menus at the top of the panel, which are used to configure the distribution. In Figure 8, reference value is set as “question”, target as “answer” and relation as “question-parentid-answer” (corresponding to r6). It means that the distribution corresponds to the cardinality constraint □0..13 at the “answer” side on r6. As the object model evolves over time, accordingly the distribution changes in different object models. The value in the “moment” drop-down menu indicates the time (represented by an event) for the presented distribution. Each bar has a cardinality number beneath it, indicating the frequency of questions that have a particular number of answers. By looking at the first three bars, one can see that approx 400 + 1000 + 450 = 1850 questions have up to 2 answers.

### B. DISCOVERY OF AOC RELATIONSHIPS

After the class model is discovered, we can mine AOC relationships based on the objects referred to by each event. The idea is that if an event refers to an object, the activity of the event refers to the class of the object. For instance, since event re1 refers to object u4, activity register refers to class user, resulting in one discovered AOC relationship between activity register and class user as shown in Figure 7.

For each AOC relationship, its cardinalities on the class side can be achieved by incorporating numbers of objects referred to by each corresponding event. Consider the cardinality on the class side of the relation between activity “register” and class “user”. Since each “register” event, i.e., re1, re2, re3 and re4, refers to precisely one “user” object, i.e., u4, u6, u157 and u7488, respectively, the discovered cardinality is 1. In contrast, the “always” cardinalities on the activity side can be achieved by incorporating numbers of events referring to each corresponding object at each moment after the object is created. Consider for example the same AOC relationship. After each “user” object is created, it is always referred to by precisely one “register” event. As a result, the discovered “always” cardinality is □1. In terms of the “eventually” cardinality on the activity side, we only check the moment when the last event happens. Since each user object is referred to by precisely one register event when the last event happens, the discovered “eventually” cardinality is □1 (omitted in the graph since it is indicated by □1).

In summary, the discovered AOC relationships between activities and classes describe the constraints between events and objects. Here, all the discovered cardinalities on the relationships are □1 and 1, which indicate the one-to-one relation between events and objects. For instance, one “register” event corresponds to one “user” object and vice versa.

### C. DISCOVERY OF BEHAVIORAL MODELS

Section V-A discovered a class model, i.e., classes and relationships between classes, while Section V-B discovered...
AOC relationships, i.e., activities which refer to classes. Based on them, we can relate events by objects and discover the behavioral constraints between activities.

Each pair of activities referring to the same class or two related classes may have potential constraints in between. The class or the relationship between the two related classes serves as the intermediary to relate events. More precisely, we take one activity as the reference activity and another one as the target activity. Based on the intermediary, we correlate target events to each reference event. If the relation between each reference event and its target event satisfies the restriction indicated by a constraint type, a behavioral constraint of this type is discovered.

Consider for example the constraint between activities register and post question in Figure 7 to understand the process of discovering behavioral constraints, based on the log in Table 3. Since (i) activity register refers to class user, (ii) activity post question refers to class question and (iii) class user is related to class question, there exist potential constraints between register and post question. By considering post question as the reference activity, there are three reference events pq1, pq2 and pq3. As shown in Figure 5, since (i) re2 refers to object u8, (ii) pq1 refers to object q1, and (iii) u8 is related to q1, we derive that re2 is related to pq1. In this way, we also derive that re2 is related to pq2 and re3 is related to pq3. In other words, each post question event precisely has one related register event happened before. This relation satisfies the semantics of constraint type unary-precedence, resulting in a discovered unary-precedence constraint between register (target activity) and post question (reference activity).

The above explanation describes the method to correlate events and discover behavioral constraints (see [11], [26] for more details). Using this method, seven behavioral constraints are discovered, as shown in Figure 7, which specify the temporal restrictions on the behavioral perspective in a declarative manner. For example, the “unary-precedence” constraint between “register” and “post_question” activities indicates that someone is only able to post questions in the website after registering himself as a user. Besides, after a question is posted, it can be answered, commented or voted. It is possible that a user gets a badge after registration.

In addition to discovering all behavioral constraints to describe the control-flow, our approach can also provide the cardinalities distribution of precedence/response target events in terms of a pair of activities, using the panel in Figure 9. The reference, target and intermediary drop-down menus in the panel indicate the reference activity, target activity and intermediary, respectively. In Figure 9 the reference activity is “register”, the target activity is “post_question” and the intermediary is “user-owneruserid-question” (corresponding to the class relation r2). By default, the distribution is for all reference events, in this case for all “register” events. It is possible to inspect the distribution for a particular reference event by setting specific instance values on the drop-down menu displayed in Figure 9.

VI. CONFORMANCE CHECKING

The discovered OCBC models describe users’ actions in social media. However, the real actions may not totally consist with the expected scenario. One can design an OCBC model or repair a discovered model based on domain knowledge, and consider it as the reference model to check conformance [13], [26].

Figure 10 presents an OCBC model derived by repairing the discovered model in Figure 7. We consider the model in Figure 10 to be the expected behavior of Stack Exchange users. Taking the model as reference, we check conformance using our plugin “Conformance Checker” in ProM and highlight the violated constraints in red in the model.

On the behavioral perspective, the “response” constraint between “register” and “get_badge” indicates that each “register” event should be followed by “get badge” events and it is highlighted due to violations for this rule. By selecting the violated constraint, the plugin can list the deviating events, e.g., “register25” and “register96”.

On the data perspective, the cardinality ♦1..* on r6 is highlighted, which indicates that some “question” objects do not have corresponding “answer” objects (it should have eventually at least one answer). Similarly, the deviating objects, e.g., “question82”, are listed in the plugin. On the interaction perspective, the relation between “close_question” and “question” is highlighted, which indicates that some “question” objects have corresponding “close question” events, i.e., these questions are closed.

^3 The plugin can be downloaded at https://www.win.tue.nl/ocbc/softwares.
The detected deviations can be investigated to derive some interesting insights for data owners. For instance, the deviating “register” events correspond to users who never get badges, indicating that they are not active in posting questions or offering answers. Some measures can be taken to encourage them to participate more, such as recommending questions for them to answer or notifying them about the advantages of getting badges. For the deviating questions, which never receive any answer, the website can invite some people to answer them, especially because having questions without answers might discourage users to join and/or to be active (which might even lead them to cancel their membership). The deviating questions which are closed indicate that these questions are of bad quality. In this situation, a warning can be sent to the corresponding user to notify him/her to improve the quality for future questions.

### VII. PERFORMANCE ANALYSIS

The performance analysis on OCBC models can be split into independent analysis on so-called correlation patterns. In real applications, it is not necessary to analyze performance for all patterns if the most relevant ones are already known. In Stack Exchange, “post_question” and “post_answer” are the most important activities, as the motivation of the website is to provide answers to questions posted by users. Therefore, the most relevant pattern is the one which has “post_question” as reference and “post_answer” as target, as shown in Figure 11(a).

Based on the pattern, we can correlate events into pattern instances for performance analysis. Figure 11(b) presents some instance examples derived by correlating events from the log in Table 3. Each instance is an event sequence containing one reference event and all target events related to the reference event. For instance, the first instance contains one reference event “pq1” and two target events “pa1” and “pa2”. Here, we take this pattern as an example to present our performance analysis result.

Figure 12 employs the dotted chart to present the performance analysis result for the example pattern. The Y axis indicates that there are almost 2,150 instances while the X axis indicates that all events in these instances happened from August, 2016 to September, 2018. Each instance corresponds to a row in the dotted chart, which contains a “post_question” event, represented by the green dot, and a set of “post_answer” events, represented by the red dots. By using the results presented by the dotted chart, we can infer some insights explained as follows:

- Questions may have long life cycles. In other words, answers are still received even after a long time from the original question creation. This is indicated by the red dots which have long distances from their corresponding green dots.
- The instances are sorted by timestamp. More precisely, the instances at the top of the dotted chart contain questions and answers posted at the moment the website was founded, while the instances at the bottom of the dotted chart contain questions and answers posted more recently. The density of the red dots in the latter instances is higher than that in the early instances, which means that the website is becoming more popular and active, since questions are answered more frequently.
- There is an explosion of questions and answers just after the website was founded—right after the website release, a quite steep green line curve and dense red dots can be noticed. Then the curve goes gently from September, 2016 and becomes steep again in the latest half year (from March, 2018 to September, 2018). The whole curve indicates: (i) the website was quite popular when it was founded; (ii) then it became less popular; and (iii) it is recovering now. This discovered insight can make more sense by combining it with other analysis such as advertisements. For instance, if the website invests the same money on advertising all the time, it is obvious that the advertisements in the middle time period were less effective.

Figure 12 employs the absolute time to present the performance result, although it is possible to view the result by other angles, as the one shown in Figure 13. This chart provides...
the performance result in the relative time, by aligning all reference events, i.e., “post_question” events, with the value 0 of the relative time. According to the aligned dotted chart, we can derive some more insights explained as follows:

- Most answers are provided in a short time period (less than one day) after the corresponding questions are posted—the dense red dots just after the green dots indicate this.
- The questions between “instance 1500” and “instance 1550” obviously received less answers than other questions. The website can investigate this situation and avoid it in the future.

In addition to dotted charts, it is possible to derive some statistics/metrics to summarize the performance of a pattern. For example, we can consider the time period from the moment that a question is posted to the moment that the last answer corresponding to the question is given as the “question duration”. By calculating the average “question duration” for all instances of a pattern, we can get the statistics to measure the performance on the pattern level. With the obtained statistics in hands, it is possible to map them onto the model, and from this step important bottlenecks could be revealed.

VIII. A CASE STUDY IN FACEBOOK

Up to now, we have explained the OCBC model discovery, conformance checking and performance analysis based on the data set from Stack Exchange. Note that these techniques can also be applied to other types of social media data. In this section we analyze a data set generated by Facebook and illustrate some discovered interesting users’ behavior patterns.

Figure 14 shows some typical events recorded by the “Activity log” in Facebook. More precisely, the orange square indicates the types of events while the blue square presents some occurred events. For instance, “Posts” corresponds to an event type and the item “Guangming Li posted something via Digg” (the second last row in the blue square) corresponds to an event of the “Posts” type. Each event has a timestamp indicating when the event happens. For instance, the timestamp in the red square indicates that the event “Guangming Li became friends with xxx” happened at 10:53 on January 27, 2019.

It is possible to download “Activity log” from Facebook which consists of a set of HTML files. Table 4 presents five example files. For instance, “your_posts” records all posts of a user, such as posting a status, sharing a link and adding a new photo. The “Example of file content” column shows one example of the content of each file. Events can be extracted from these files, since timestamps are included in the file content. For instance, a “send request” event happened at 15:55 on May 18, 2017 can be extracted based on the content example in the “sent_friend_requests” file.

Based on an event log extracted from the data shown in Table 4, an OCBC model can be discovered as shown in Figure 15. The model describes an interesting behavior pattern, which is illustrated as follows. A stranger refers to a person that is not the user’s friend yet, but is the friend of a friend. After the stranger creates a post and one friend of the user comments on it (indicated by \( c_1 \)), the post will appear in the user’s “News Feed”. If the user is interested in the post, he/she can reply it (indicated by \( c_2 \)). After replying the post, the user eventually sends a friend request to the stranger and they end up being friends (indicated by \( c_3 \)). The discovered pattern can be extracted from a user’s activity log (as shown in Figure 14), i.e., this user tends to send friend requests to people whose posts are commented by the user’s current friends.

This behavior pattern might be valuable for Facebook. For instance, suppose that there is a reason to connect this user to any of his/her friend’s friends, then Facebook can starting showing more posts like this in his/her News Feed. In addition, the user perception is that he/she has more friends
now and, as consequence, Facebook becomes a more active social media platform. Besides, the friends serving as bridges (connecting the user to strangers) have a big influence on the user, maybe because they have the same hobbies as the user. Therefore, some advertisements targeted to these people can also be sent to the user.

Figure 16 presents two other discovered behavior patterns. The number in parenthesis on each behavioral constraint indicates the degree the constraint is supported in the data. In Figure 16(a), $c_1$ indicates that 78% of the user’s posts are commented by friends while $c_2$ means that the user replies to 90% of the received comments. According to this pattern, we can infer that the user’s posts are interesting and attractive to his/her friends. Besides, the user appreciates friends’ comments by almost replying to each of them. In Figure 16(b), $c_1$ indicates that the user comments on friends’ posts actively (70%) while $c_2$ means that in most cases the user comments on a post before sharing it. According to this pattern, we can infer that the user is quite active in Facebook. Moreover, the user is responsible for sharing posts, as he/she often reads the post and comments on it first.
These two patterns can be used to evaluate a user’s liveness and the quality of a user’s posts and shared posts in Facebook.

IX. RELATED WORK

Mining social media is a burgeoning multidisciplinary area where researchers of different backgrounds can make important contributions. In this section, we review the existing approaches for analyzing social media data.

A. DATA MINING APPROACHES

Data mining techniques have been widely used in social media, especially in several representative areas. For instance, community analysis [2], [21] can detect a community which is formed by individuals such that those within a group interact with each other more frequently than with those outside the group. Besides, topic detection and monitoring help to understand what topics are popular in social media, providing insights into product sales, political views, and future social attention areas [4]. Moreover, sentiment analysis and opinion mining allow businesses to understand product sentiments, brand perception, new product perception, and reputation management [14], [19]. The data mining techniques focus more on users and contents and the way these are connected as a network, i.e., who is social media, what is social media and how is social media. They overlook the events in the data, i.e., when is social media.

B. PROCESS MINING APPROACHES

Various process mining techniques have been developed for flexible processes (e.g., healthcare processes) similar to the processes in social media. Declare Miner [17] can discover Declare models which describe flexible processes in a declarative manner. References [7], [16], [18] demonstrate several approaches to discover artifact-centric models for processes with one-to-many and many-to-many relations. Reference [22] proposed a method to discover frequent behavioral patterns, named local process models, for the unstructured processes. Event logs may log performers, e.g., the person completing some activity (like the users in social media) and support for organizational mining [15]. References [20], [27] proposed techniques to discover interactions among coworkers using this performer information in event logs. They build a social network based on the handover of work from one performer to the next. Besides, they construct a sociogram which can be used to analyze interpersonal relationships in an organization. Traditional process mining techniques mostly focus on the structured control-flow perspective and they have never been applied to social media data based on our knowledge. The existing techniques mentioned above fail to discover models with a strong data perspective to describe the data structure, from data without case or artifact notions.

X. CONCLUSION AND FUTURE WORK

In this paper, we applied a novel family of process mining techniques, i.e., OCBC techniques, to real life data from social media, such as Facebook and Stack Exchange. The experiment shows that we can (i) discover the users’ behavior patterns, e.g., two patterns involving post, comment, reply and share activities in Facebook, (ii) detect deviating and/or undesired behavior, e.g., a question is posted without any answer in Stack Exchange, and (iii) provide useful insights on the time perspective, e.g., most answers are given in one day after the corresponding questions are posted.

Process mining techniques can discover process-related insights, which can compensate data mining techniques on the time aspect. In comparison, our approaches can discover more complex users’ behavior patterns (involving multiple activities, classes and interactions between them) rather than simple association rules. Besides, conformance and performance can be analyzed to detect deviations and bottlenecks.

In future, we can combine process mining and data mining techniques to solve problems which cannot be done only with either of them. For instance, in the Stack Exchange website, we can evaluate the quality of an answer by using the process-related and time-related insights (e.g., considering the time after the corresponding question is posted and if the question receives “vote up” or “vote down” events before or after the answer) discovered by process mining and content-related insights (e.g., to what extent the contents of the answers match the corresponding questions) discovered by data (text) mining.

REFERENCES


