Proceedings of the Federated Conference on Computer Science and Information Systems pp. 341–348
DOI: 10.15439/2018F325
ISSN 2300-5963 ACSIS, Vol. 15

Mining e-mail message sequences from log data

Paweł Weichbroth
Faculty of Management, WSB University in Gdansk,
Grunwaldzka 238A, 80-266 Gdansk, Poland
pawel.weichbroth@hotmail.com

***Abstract*—Communication by electronic mail (e-mail), once extravagant, is now the usual way to exchange data and information. Widely accepted by Internet users, business and governments, it is claimed to be the key part of the e-revolution. E-mail systems have been successfully implemented in almost all computer-aided domains of human interest, providing efficient, effective and permanent mechanisms of transmission. However, to date, the capability to exhibit an ordered list (sequence) of e-mail message senders and recipients, with the respective duration time between receiving and answering is still lacking. To fill this gap, in this paper we introduce the SOMF algorithm for mining such sequences from server log data. We specified a three-stage approach to comprehensively target the problem. The first stage concerns a data preparation task in order to assemble the input for the algorithm. The second, known as data mining, is the automatic analysis of data input performed in an unsupervised model by the SOMF algorithm. The third embraces output (knowledge) visualization, interpretation and evaluation. The given case study is based on the log data from an operational STMP server. By design, this simplified example brings about a better understanding of the solution, indicating one of its potential applications to identify and eliminate deadlocks in the realization of business processes. We also tested the efficiency of the implementation of the algorithm in five independent experiments on seven datasets, ranging in size. The results show that mining even 1 million rows is performed in approximately less than 6 minutes.**

I. Introduction

IN THE last decade, we have observed that the approach to information system design is evidently shifting from data to processes. Effective identification, construction, evaluation and deployment of mission-critical processes can give a competitive and strategic advantage to an organization [1]. Some argue that knowledge is still the most important asset to influence overall performance and the ability to innovate [2], [3], [4].

Knowledge workers utilize information technologies and their productivity depends on particular computer applications. The McKinsey Global Institute (MGI), through interviews with 4200 managers from companies in different businesses, in July 2012, reported that 28 percent of total work includes time reading, writing or responding to e-mails [5]. In 2015, the number of business e-mails sent and received per business user per day totalled 122 e-mails, and by the end of 2019 is expected to average 126 messages [6]. Today, there is rising concern that for some workers the volume of e-mail has grown to the size which in turn has negative effects on well-being and performance [7]. The perception of an individual being unable to find, organize or process his/her e-mails effectively is defined as the feeling of e-mail overload [8].

Knowledge management has long been a valuable part of business process architectures and models of various complex and collaborative domains [9]. A business process can be defined as a sequence of activities located and bounded in the frame of a particular organization [10], describing steps and corresponding tasks assigned and performed by particular participants (humans or other physical beings), the objective of which is to achieve a desired result [11]. Speaking from the margins, Adam Smith's theory of labour division is still up-to-date.

Business process design presents assumptions and beliefs in compliance with specified goals [12]. However, in real-life scenarios, an empirical business process can suffer from the following burdens: (1) some participants may not respect assumptions, their roles or tasks, and in consequence, act in a different way; on the other hand, even if they do, (2) some of them may delay performing assigned tasks, intentionally or not (for instance due to e-mail overload). Thus, one can ask for an objective method that provides data-driven evidence that shows how the process is empirically achieved.

In this paper, we investigate one particular activity which concerns e-mail correspondence between participants, and the primary focus is to introduce an algorithm, namely SOMF, for discovering e-mail message sequences from server log data. In this narrow extent, the obtained knowledge can be used for conformance checking, which aims to detect inconsistencies between the model of processes and their corresponding execution. In particular, it can be a way to detect communicated burdens on the one hand, and act as a starting point to discuss possible improvements on the other.

We devote our contribution in the field of data mining algorithms, to extracting sequences from data. From a broader perspective, the elaborated approach can be seen as an autonomous component of competitive intelligence [13], which, being a strategic tool producing actionable intelligence, in turn supports organizations in the decision-making process [14], improves their performance [15], and eventually fosters a competitive advantage [16].

The rest of the paper is organized as follows. The next section provides the problem statement for mining sequences from data. In Section III, the knowledge discovery process is outlined and specified in the frame of the research agenda, and divided into three subsections. In the next section, the process is exemplified by the case study. The last section closes the paper by presenting and discussing the obtained results from testing the efficiency of the algorithm.

II. The Problem Statement

As stated in the previous section, this research study introduces the concept of mining e-mail message sequences from server log data. The problem can be epitomised by three main aspects: (1) data preparation, (2) data mining, and (3) knowledge visualization, interpretation and evaluation.

This study mainly focuses on solving the first and second, by explicitly formulating algorithms devoted to each one. To achieve this goal, firstly, we formulate and provide all the relevant definitions.

Definition 1. A dataset $D = \{t_1, t_2, \ldots, t_n\}$ is a set of transactions, where each transaction is described by four attributes: message-id, time, sender and recipient.

Definition 2. A message m_i is an abstract term that may represent a document or e-mail uniquely identified by the message-id.

Definition 3. The time of the transaction tt_i is the recorded execution time of the performed action by the sender.

Definition 4. A set $P = \{p_1, p_2, \ldots, p_m\}$ is the finite collection of participants, which can be both message senders or recipients.

Definition 5. An event *e* is a pair $(x \rightarrow y)$ of the sender *x* and the participant *y*, where $x \neq y$.

Definition 6. A weight w_i is the difference between execution times tt_i and tt_{i-1} of two subsequent transactions, where $w_1 = 0$, and for $i = 1, 2, \ldots, n$.

Definition 7. A sequence $s \leq (e_1) : w_1, (e_2) : w_2, \ldots, (e_k) : w_k)$ of events is an ordered list of nonempty events; for each integer $k = 1, 2, \ldots, n$; a sequence of the length k is called a k-sequence.

Definition 8. A directed, weighted and labelled graph G is a tuple (V, E, w) consisting of a finite set V, together with a subset $E \subseteq VxVxR$. The elements of V are the vertices of the graph, and the elements of E are the arrows of the graph. An arrow of a graph is an ordered pair $[x, y]$, where x and y are the vertices of the graph, and w is the associated real number of the pair, called its weight, where $x \neq y$. The labels for vertices are a subset L of P.

To visualize knowledge, a user can use any application capable of processing data, which as a result, displays the adequate drawing. The remaining two tasks are usually associated with the specific context of the problem domain, and therefore should not be generalized.

III. The Knowledge Discovery Process

In our approach, the process of knowledge discovery is divided into three stages, followed one by one and independently performed ($1 \rightarrow 2 \rightarrow 3$), in a similar way to well-recognized and accepted models, such as KDD or CRISP-DM.

A. Data preparation

The data preparation stage concerns sorting objects. In the first step, a new message list is initiated. Next, a data set D is scanned, and the total number of rows n is determined. In the body of the loop $(4-6)$, where n is the termination condition, four attributes are selected from each row and form a new object, inserted into the message list. Next, a message list is grouped by the message-id, and sorted in ascending order by a time stamp. The corresponding pseudocode is given below.

Input: D

```
1 Initiate New List(Message-List);
2 Read(D);
3 for i := 1 to n
4  select MessageId, Time, Sender, Recipient from D;
5  create Object(Object-Message);
6  insert to List(Message-List);
7 end;
8 group List(Message-List) by MessageId and
9 sort ASC List(Message-List) by Time;
```

Output: a Message-List.

B. Data mining

The message list is now the input with no parameters for the SOMF algorithm. The pseudocode below shows the main idea lying behind its construction.

Input: List(Message-List)

```
1  while i < Count List(Message-List) do
2   CheckMessage:= read Object.MessageId[i];
3   MessageTime:= read Object.Time[i];
4   for each unique object from List(Message-List)
5    MessageTime:= read Object.Time[i];
6     add vertex(sender) to Sender List(Adjacency-List[0]);
7     add vertex(sender) to Sender List(Adjacency-List[1]);
8     weight:= (MessageTime[i] – MessageTime[1]);
9     add vertex(recipient) & weight to Recipient List(Adjacency-List[0]);
10  end;
11  for each object from List(Message-List)
12   if vertex(sender) exists in List(Adjacency-List[i]) then
13    weight:= (MessageTime[i] – MessageTime[1])
14    add vertex(recipient) & weight to Recipient-List(Adjacency-List[i+1]);
15   if vertex(sender) does not exist in List(Adjacency-List[i] then
16    add vertex(sender) to Sender-List(Adjacency-List[i+1]) and
17    weight:= (MessageTime[i] – MessageTime[1])
18    add vertex(recipient) & weight to Recipient List(Adjacency-List[i+1]);
19   if vertex(recipient) does not exist in List(Adjacency-List[i]) then
20    add vertex(recipient) to Sender-List(Adjacency-List[i+1]);
21   end;
22  if Object.MessageId[i] ≠ Object.MessageId[i+1] then
23     create Graph(MessageId);
24  i++
25 end.
```

Output: A set of graphs.

In the body of the first loop (2–9), for each unique object from the message list, the time of the first sent message is determined; next, a new vertex, representing the message sender, is created and added to the adjacency list on the left side; a new vertex, representing a message recipient, is created and added to the adjacency list on the left side; the weight between the top vertex and the vertices one level down equals zero.

In the body of the second loop (12–22), if the vertex of the sender exists in the adjacency list, then the weight is calculated, and a new vertex, representing a recipient, is created and added along with the weight to the adjacency list on the right side; if the vertex representing the sender does not exist, then a new vertex is created and added to the adjacency list on the left side; the weight is calculated and a new vertex for each recipient is created and added along with the weight to the adjacency list on the right side; if the vertex representing a recipient does not exist, then a new vertex for each recipient is added to the adjacency list on the left side. Finally, if the identifier of the next object is different, then a graph representing a unique e-mail message sequence is created.

We used an adjacency list as the graph representation. This was the most suitable form for us to use. Our goal is neither to prove its appropriateness or efficiency nor to investigate different representations. Having said that, however, if we take into account the results obtained from implementation feasibility and performance testing of the algorithm, the correctness of choice should not be a subject for long discussion.

C. Knowledge visualization, interpretation and evaluation

Knowledge visualization aims to use visual representation to facilitate the understanding of discovered complex data structures [17]. Knowledge interpretation is an arbitrary construct of the information perceived by an individual who aims to specify its meaning by incorporating context, logic and experience [18]. Knowledge evaluation is a subjective judgment on its applicability and validity to solve a particular problem [19]. These three tasks are wider discussed in the next section, having the input, i.e. visualized knowledge, already generated.

IV. Case Study

Let us consider a dataset $D = \{t_1, t_2, t_3, t_4, t_5\}$ of five transactions and a dataset $U = \{a, b, c, d, e\}$ of five users that are both senders and recipients, whose e-mail account names equal their names in the *gdansk.com* domain. Let $M = \{m_1\}$ be a set of one message m_1, represented by the unique identifier $m.1$. Each transaction is described by four attributes, i.e. *message-id, time stamp, sender* and *recipient*. The first transaction t_1 of the input for the data preparation stage is shown below.

```
Message-ID: <m.1@gdansk.com> Date: Fri, 19 Aug 2016
11:30:00 From: =?ISO-8859-2?Q?a?=<a@gdansk.com> To:
=?ISO-8859-2?Q?b?= <b@gdansk.com>
```

To simplify the log data, on the same day, in the one message domain, for the first user, it takes 5 minutes to pass over the message, and for each subsequent user, five minutes more. The message list is depicted in Table 1. Each transaction can be interpreted analogously to the first one: "at 11:30 a.m. the user a (sender) sends the e-mail to the user b (recipient)".

In the second stage, the algorithm scans the message list, and for the first transaction t_1 determines the execution time of the message sent; next, the sender vertex a and recipient vertex b are added respectively to the first and second position on the left side of the adjacency list; only for t_1 the weight is not calculated and equals zero; the sender vertex b is added to the first position on the right side of the adjacency list and the weight (0) is assigned. The sender vertex b exists on the list; the weight is calculated and the vertex recipient c is added with the weight (5) assigned. The sender vertex c is added to the third position on the left side of the adjacency list; the weight is calculated, and vertices d and e are added on the right side with the weight (10) assigned. The sender vertex e is added to the fourth position on the left side; the weight is calculated and the recipient vertex a is added on the right side with the weight (15) assigned. The sender vertex a exists, the weight is calculated and the recipient vertex e is added as the second on the right side with the weight (20) assigned.

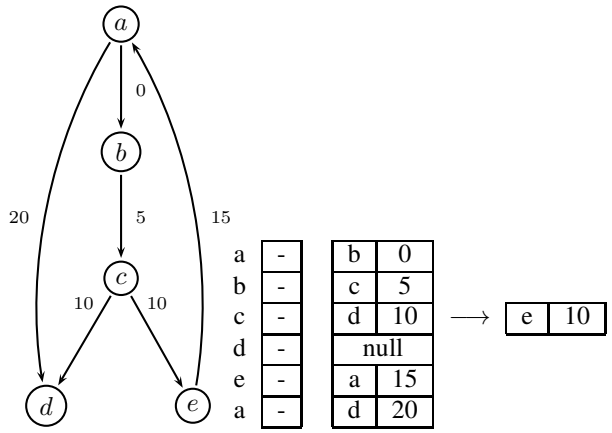

Figure 1. The graph G and the adjacency list

The discovered graph G can be interpreted literally as the following sequence of events:

- (e_1): at a given time, a message was sent from participant a to b;
- (e_2): five minutes later, a message was sent from participant b to c;
- (e_3): ten minutes later, a message was sent from participant c to d and e;
- (e_4): fifteen minutes later, a message was sent from participant e to a;
- (e_5): twenty minutes later, a message was sent from participant a to d;

TABLE I. THE MESSAGE LIST

transaction (t_i)	**time stamp** (hh:mm)	**sender** (name)	**recipient** (name)
t_1	11:30	a	b
t_2	11:35	b	c
t_3	11:45	c	d, e
t_4	12:00	e	a
t_5	12:20	a	d

TABLE II. THE ALGORITHM EFFICIENCY ACCORDING TO DATASET SIZE

D no. of rows	**file size** (KB)	**test1** (mm:ss.ms)	**test2** (mm:ss.ms)	**test3** (mm:ss.ms)	**test4** (mm:ss.ms)	**test5** (mm:ss.ms)
10	2	01.00	01.00	01.03	01.01	01.01
100	18	01.01	01.01	01.01	01.02	01.01
1 000	182	01.06	01.06	01.06	01.06	01.07
10 000	1827	01.59	01.59	01.59	01.59	01.61
100 000	18262	07.52	07.44	07.43	07.50	07.51
500 000	91309	38.96	46.19	40.30	38.99	39.49
1 000 000	182618	05:51.89	05:32.95	05:32.75	05:32.95	05:24.26

V. EFFICIENCY EVALUATION

The SOMF algorithm has been implemented in C#. Firstly, to verify its correctness, we prepared and used a dataset in such a manner that allowed us in advance to determine the output. Secondly, we implemented and executed a stand-alone script which randomly generated seven datasets, ranging in size from 10 to 1 million records. Finally, we separately performed five tests on each dataset, using a mobile computer equipped with an Intel Core I5 (3230M @ 2,6 GHz) processor, 4 GB (DDR3) RAM, and Microsoft Windows 8.1 (x64). The obtained results are summarized in Table 2.

If we take into account only the first four datasets, then the mining duration does not exceed 2 seconds and is comparably the same. Differences can be noticed in the last two datasets; however, the standard deviation is again relatively low, respectively 2,87 and 8,95 seconds. Now, if we consider the largest dataset, such a total number of rows cannot represent one hypothetical message sequence because it is simply too complex to be realized in any real-life scenario. Yet, conversely, in our opinion, the duration of less than 6 minutes is relatively low, if compared to other typical domains of the application of data mining algorithms. To sum up, the algorithm efficiency is at an acceptable level.

REFERENCES

[1] L. Mancilla-Amaya, C. Sanin, C., and E. Szczerbicki, "Using Human Behavior to Develop Knowledge-Based Virtual Organizations". *Cybernetics and Systems: An International Journal*, 41(8), pp. 577–591, 2010.

[2] M. Owoc, and K. Marciniak, "Knowledge management as foundation of smart university". Federated Conference on Computer Science and Information Systems. IEEE, pp. 1267–1272, 2013.

[3] M. Hernes, "Knowledge Integration Method for Supply Chain Management Module in a Cognitive Integrated Management Information System". In: International Conference on Computational Collective Intelligence, pp. 81–89. Springer, 2016.

[4] M. Pondel, and J. Korczak, "A view on the methodology of analysis and exploration of marketing data", 2017 Federated Conference on Computer Science and Information Systems. IEEE, pp. 1135–1143, 2017.

[5] M. Chui et al., "The social economy: Unlocking value and productivity through social technologies". McKinsey Global Institute, pp. 46, 2012.

[6] The Radicati Group. A Technology Market Research Firm. "Email Statistics Report 2015-2019", p.3. London (UK) 2015.

[7] K. Reinke, and T. Chamorro-Premuzic, "When email use gets out of control: Understanding the relationship between personality and email overload and their impact on burnout and work engagement". Computers in Human Behavior, 36, pp. 502–509, 2014.

[8] L. A. Dabbish, and R. E. Kraut, "Email overload at work: An analysis of factors associated with email strain". In Proceedings of the ACM conference on computer supported cooperative work (CSCW), pp. 431–440. New York, ACM Press 2006.

[9] P. Wang, C. Sanin, and E. Szczerbicki, "Prediction based on integration of decisional DNA and a feature selection algorithm RELIEF-F". *Cybernetics and Systems*, 44(2–3), pp. 173–183, 2013.

[10] A. Przybyłek, "The Integration of Functional Decomposition with UML Notation in Business Process Modelling". In: Advances in Information Systems Development, pp. 85–99. Springer 2007.

[11] B. Marcinkowski, and M. Kuciapski, "A business process modeling notation extension for risk handling". In: A. Cortesi, N. Chaki, K. Saeed, and S. Wierzchoń (Eds): Computer Information Systems and Industrial Management, pp. 374–381, Springer 2012.

[12] A. Przybyłek, "A Business-Oriented Approach to Requirements Elicitation". In: 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'14), Lisbon, Portugal, 2014.

[13] R. Pellissier, and T. E. Nenzhelele, "Towards a universal competitive intelligence process model". South African Journal of Information Management, 15(2), 1–7, 2013.

[14] D. Heppes, and A. Du Toit, "Level of maturity of the competitive intelligence function: Case study of a retail bank in South Africa". Aslib Proceedings: New Information Perspectives 61(1), 48–66, 2009.

[15] B. Huijbrechts, M. Velikova, S. Michels, and R. Scheepens, "Metis1: An integrated reference architecture for addressing uncertainty in decision-support systems". Procedia Computer Science, 44, 476–485, 2015.

[16] R. Brody, "Issues in defining competitive intelligence: An exploration", Journal of Competitive Intelligence and Management 4(3), 3–16, 2008.

[17] J. Korczak, H. Dudycz, and M. Dyczkowski, "Design of financial knowledge in dashboard for SME managers". In: Computer Science and Information Systems, pp. 1123–1130, IEEE 2013.

[18] M. Owoc, P. Weichbroth, and K. Żuralski, "Towards better understanding of context-aware knowledge transformation". In: Computer Science and Information Systems, pp. 1123–1126, IEEE 2017.

[19] M. L. Owoc, "Wartościowanie wiedzy w inteligentnych systemach wspomagających zarządzanie". Prace Naukowe Akademii Ekonomicznej we Wrocławiu. Seria: Monografie i Opracowania. Wrocław 2004.