Emails Analysis for Business Process Discovery

Nassim LAGA[1], Marwa ELLEUCH[1,2], Walid GAALOUL[2], and Oumaima ALAOUI ISMAILI[1]

[1] Orange Labs, France, `{FirstName}.{LastName}@orange.com`
[2] Telecom SudParis, France, `{FirstName}.{LastName}@telecom-sudparis.eu`

Abstract. Most often, process mining consists of discovering models of actual processes from structured event logs. However, some business processes (BP), or at least some parts of them, are not necessary supported by an information system (IS), and consequently do not leave any structured events log. Therefore, applying traditional process mining techniques would generate a partial view of such processes. Process actors often rely on communication tools to collaboratively execute their business processes in such situations. However, given the unstructured nature of communication tools traces, process mining techniques could not be applied directly; thus it is necessary to generate structured event logs by recognizing the process-related items (activities, actors, instances, etc.). In this paper, we address this challenge in order to mine business processes from email exchange traces. We introduce an approach that minimizes users' efforts to manage the growing amounts of exchanged emails: It enables to *collaboratively*, and *gradually* build an annotated corpus of messages, and to *automatically* classify these ones into process, instance and activity IDs using machine learning techniques. Compared to related works, we facilitate the task of obtaining annotated datasets and we investigate the use of email exchange histories, correspondent, references and named entities for building clustering and classification features. The proposed approach is evaluated through a proof of concept and successfully experimented on an email dataset.

Keywords: Process mining · Business process management · Clustering · Supervised learning · Named entities.

1 Introduction

Process mining consists in extracting useful knowledge from event logs generated by a variety of software tools hosted in Information Systems (IS) [21,23]. It enables business experts to discover new processes, new practices, as well as BP limitations. However, most of them assume that: (1) Event logs have a **structured** format, and (2) the business process is **totally executed in IS**, and consequently event logs contain the trace of **all** executed business tasks. However, several business activities could be achieved using informal methods, such as communication tools (e.g. email exchange, IM, etc.). As a consequence, the traces of these activities are not present in traditional event logs. In addition, they are often not structured. One of the important tasks to be handled when starting from unstructured log data to mine processes is how to convert it into structured event logs, which is compatible with the available process mining techniques. The task of constructing structured event log starting from unstructured log data of a given trace of communication consists mainly in recognizing the process-related items (name, activity and instance).

Up until recently, only few studies have properly addressed the recognition of all these related information. Most of them focus on the unstructured email exchange traces to mine business processes by using learning [8,4,17,10,7,9] or pattern matching [11,1] techniques. These proposals suffer from the following limitations. Firstly, they require a considerable **human intervention** with time consuming tasks. In the case of pattern matching based approaches, patterns are defined manually. As for the supervised learning based approaches, a huge human effort is required for building a training dataset: In most cases, a big amount of unorganized and unlabeled data has to be manually annotated. Even if unsupervised learning techniques can be introduced as a possible alternative to avoid preparing such a corpora [8,9], some manual tasks are still needed; they consist in labeling or modifying (correcting) the generated clusters and tuning the parameters of some parametric algorithms such as kmean and hierarchical algorithms. Second, they tend to generate **unreliable business process models**. This can be the result of: (1) Relying only on clustering techniques, which is error prone [8,9], and (2) using features that are not discriminative enough to recognize some business process related information [4,17,10,7]. Discriminative features are the most relevant variables for clustering. These latter depend on the type of the knowledge that we want to extract. In the case of process mining, existing works mostly exploit the entire content of the unstructured email related data (subject or email body) to build some of their learning features [12,8,9]. These features are used then for all kind of BP knowledge extraction tasks. Typically, the textual data of emails may contain key terms which can help to separate emails according to their BP for example. However, given emails belonging to the same BP but to different instances, their textual data are likely to share the same BP key terms, which means that using them entirely probably increases the confusion between instance clusters. On the other hand, named entities (e.g. person, company names) and references (e.g. customer reference, product reference) differ probably from one instance to another even if they belong to the same BP, which means that they can have a considerable contribution in separating emails into instances.

In this paper, we address these challenges in order to mine business processes from email logs. We consider that human intervention is often required to generate reliable business process models but we aim to minimize it. We propose then an approach that enables users to *collaboratively* and *gradually* build an annotated corpus of messages and to automatically classify these ones into process, instance and activity IDs using machine learning techniques. Our proposal differs from existing works by: (1) Reducing the human effort required for obtaining an annotated dataset through a collaborative and progressively learning approach (2) Investigating the use of email exchange histories, its participant correspondent entities, references and named entities existing in its body for building clustering and classification features (3) Applying a fast and non-parametric clustering algorithm for process instance detection.

The rest of the paper is organized as follows. First, we introduce in section II an overview of our contributions, the overall algorithm, and functional entities. Then, we detail them in section III. In section IV, we validate the proposals by a proof of concept and its application to detect some processes (e.g. hiring process and patent application process) taken from real data. We discuss the related work in section V and conclude with a summary and perspectives in section VI.

2 Overview

Our proposal combines classification and clustering techniques for mining process models from email exchange traces. To achieve this task, a structured event log which contains process, activity and process instance labels must be generated. In this paper, we assume that each email is related to one activity, one process, and one instance. Our approach is summarized in Figure 1 and is a sequential combination of the following steps:

Step 1: Process and activity labels generation: This step is initially done manually and collaboratively by users. Then, a predictive model is trained gradually with the obtained annotated data for recognizing process and activity names. The classification features, which are used in the training phase, are built from the following email parts: subject, content, historical exchange and correspondent entities of email participants. Once reaching reliable prediction performances, the task of process and activity labels generation will be automatically performed by the obtained predictive models.

Step 2: Process instance detection: The purpose of this step is to detect the process instance related to each email. A clustering algorithm is applied at this step. The used distance matrix is defined as a weighted sum of sub-distances related to the following email parts: (1) time, (2) correspondent entities of email participants, and (3) content and subject reduced into references and named entities.

Step 3: Event log generation: The goal is to generate time-ordered per-process event sets. Each event presents an email with its timestamp and its activity, process and instance ID labels.

Step 4: Process model discovery: Any process discovery algorithm can be applied here to mine the business process models.

3 Approach

3.1 Step1: Process and activiy labels generation

The goal of this step is to associate process and activity labels to new incoming emails. More formally, it is a function $F : EM \to P \times A$ where EM denotes a set of emails e, P presents a set of process names and A presents a set of activities.

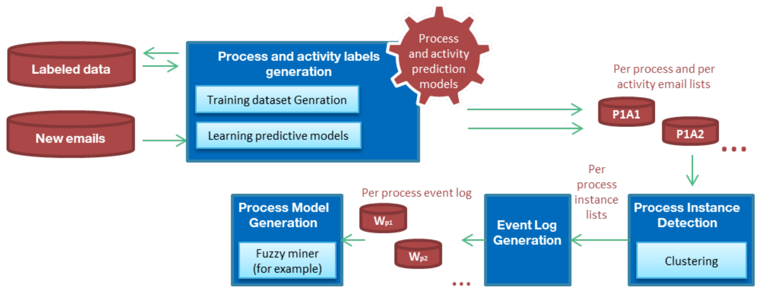
Labeled data
New emails
Process and activity labels generation
Training dataset Genration
Learning predictive models
Process and activity prediction models
Per process and per activity email lists
P1A1
P1A2
Per process instance lists
Process Instance Detection
Clustering
Event Log Generation
Per process event log
Wp1
Wp2
Process Model Generation
Fuzzy miner (for example)

Fig. 1. Process Mining From Email logs: Main Steps

Algorithm 1 Process and activity labeling of one incoming email

```
 1: procedure TAGEMAIL(
        $e, aEL, pEL, pET, emC, erC, clfs, minBatch$)                                  ▷
      ▷ e: the email to be annotated
      ▷ aEL: the list of annotated emails
      ▷ pEL: the pseudo event log
      ▷ pET: the error rate threshold
      ▷ emC: the email count since model train (It is initialized outside the procedure)
      ▷ erC: the errors count since model train (It is initialized outside the procedure)
      ▷ clfs: the classifiers
      ▷ minB: the minimum size of the new data to train new models

 2:     $p \leftarrow False$                                  ▷ will contain the predicted process
 3:     $a \leftarrow False$                                  ▷ will contain the predicted activity
 4:     $uP \leftarrow False$                                 ▷ will contain the user given process
 5:     $uA \leftarrow False$                                 ▷ will contain the user given activity
 6:     $emC \leftarrow emC + 1$
 7:     if $clfs$ then
 8:         $p, a \leftarrow PredictLabels(e, clfs)$
 9:     $uP, uA \leftarrow ManualTagging(e)$
10:     if $uP$ AND $uA$ then
11:         $erC \leftarrow erC + 1$
12:         $p \leftarrow uP$
13:         $a \leftarrow uA$
14:         $aEL \leftarrow aEL \cup \{(e, uP, uA)\}$
15:         if $erC/emC > pET$ AND $emC > minBatch$ then
16:             $clfs \leftarrow TrainPredictiveModels(aEL)$
17:             $emC \leftarrow 0$
18:             $erC \leftarrow 0$
19:     if $p$ AND $a$ then
20:         $pEL \leftarrow newEvent(pEL, e, p, a)$
21:     return $p, a, erC, emC, aEL, clfs, pEL$
```

Each email e in EM is defined as 7-uplet : FROM (email sender), TOs (email recipients), CCs (emails addresses that are in copy of the email), SUB (email subject), CONT (email content), Timestamp, HistExch (The content of historical exchange of an email). This task is done manually and collaboratively by users until reaching reliable predictive models. Then, it will be performed automatically by these models which are trained using Mini-batch learning approach. This approach consists on waiting until obtaining a batch of new observations and then train the already existing models on this whole batch. The pseudo code of this step is described in Algorithm 1.

First, the associated process and activity names for each email e are predicted using existing models (line 8). Then, if manual annotations (process and activity names) are given by the user (this could happen if the annotations are not, or wrongly, predicted) errors rate is checked, if it is higher than a threshold (pET) and a minimum size of data is collected (minB), we train again the models (process and activity classifiers) (lines 10→16). Finally, the predicted, or corrected, process and activity names, associated to the email, are saved in the pseudo event log (pEL) dataset (line 20).

For learning or updating predictive models, we have to dispose a training dataset that must be converted into a (X,Y) couple, where X is a matrix having in each row the feature values of each sample and Y is a vector representing the corresponding targets. We further detail in the following sections our classification features and preprocessing steps applied to generate this matrix X.

a) Defining classification features: We split here the task of selecting the efficient attributes for building our classification features according to the prediction task type. To predict activities, we use the correspondent entities of email participants (FROM, TOs, and CCs) and the email subjects (SUB) and we add the content parts because correspondent and subject parts are not sufficient enough to recognize activities since they lack precision about them. We have handled also the case where the email contains one short sentence. This type of email can be sent, for example, to confirm or deny what has been said in previous emails. In this case, we use also the email exchange history (HistExch) because it is not obvious to understand the business goal of sending such emails without analyzing the history of the concerned discussions. To recognize Business processes, we use only the correspondent entities and the subject parts because we noticed that content parts degrade the recognition accuracy when they are introduced with the same preprocessing steps as in the activity prediction phase.

b) Defining preprocessing steps: We summarize these steps as follows:
Replace Particular Expressions: We detect, using regular expressions, special expression within the subject (and the body in the case of activity prediction) (e.g. HTTP links, phone numbers, special references if known, and file names and their extension), and replace these expressions with a special tag (e.g. HTTP_LINK, PHONE_NUMBER, IS_REF, and PPT_FILENAME etc.).
Remove Person Names: We detect all the correspondents' first names and last names, and then remove them from the textual data.
Lemmatize The Text: This method reduces the dimension of the resulting matrix (Eq (2)). It consists in reducing the different forms of word to a single form (e.g. words "thinks", and "thinking" into one single form "think").
Remove Stop Words: This function is useful to remove the most common words in the emails that can be distracting and non-informative. We used the nltk[3] library, enriched with additional words detected during the data exploration step (e.g. regards, hello, outlook prefixes, etc.).
Generate 1-gram,2-gram Vocabulary: We first split the remaining text into a list of words and detect the different sequential combinations where each item

[3] https://www.nltk.org

has the size of 1 to 2 words (1-gram, 2-gram). Then, we remove the most and the less frequent terms to improve generalization of our predictive models. In fact, sparse terms can generate wrong associations and overly common words don't present relevant information to differentiate between email intents.
Generate TFIDF (Term frequency-inverse document frequency): This function generates a TFIDF matrix which encodes the frequency of 1-gram and 2-gram terms in an email with respect to the rest of the corpus. The size of this matrix is equal to (N, T) where N is the total number of emails and T is the total number of different 1-gram and 2-gram terms.
Generate Interaction Matrix: In order to emphasize the contribution of the email correspondent entities in defining process and activity names, we build a presence matrix (PM) reflecting for each email the interlocutors' entities (sender entity, recipient entity, and copied entity).

Definition 1 *Let E be the set of emails of length N, L be the list of different entities of length M, C be a list where each element corresponds to the list of participant correspondents' entities of each email, PM be a matrix whose columns represent L and rows represent E. PM can be defined as follows:*

$$PM(i,j)_{i,j\in[0,N-1]\times[0,M-1]} = \begin{cases} 1 & \text{if } L[j] \in C[i] \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

GenerateTheFinalMatrix: This function generates the X matrix which is a weighted concatenation of PM and TFIDF matrices.

$$X = \left[\beta_1 TFIDF_{(N\times T)}\ \beta_2 PM_{(N\times M)}\right] \tag{2}$$

Where the weights β_1 and β_2 will be defined empirically.

3.2 Step2: Process Instance Detection

The goal of this step is to detect a specific occurrence or execution of the same business process, which is known as process instance. We use the *pseudoEventLog* dataset (*pEL*) generated as an output of step 1 which contains the list of emails correctly annotated with corresponding process and activity names. We first divide it into per-process groups. Then, we apply a clustering technique on each obtained group to detect clusters corresponding to process instances.

Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) should have similar properties or features, while objects in different groups should have highly dissimilar ones. A clustering algorithm takes as input a similarity matrix M which defines the similarity between each couple of emails. This matrix is formally specified in Definition 2

Definition 2 *Let N be the total number of emails, E be the set of emails in our corpus and $f : E \times E \to R$ be the similarity function that calculates a similarity value between two emails. The similarity matrix M can be defined as a square matrix of size $N \times N$ where $M[i,j]_{i,j\in[1,N]} = f(E[i], E[j])$*

In our case, the similarity function f is a distance function defined by Eq(3).

Our clustering phase goes mainly through the following sub-steps: (1) Identify clustering features. (2) Generate the similarity matrix. (3)Apply a clustering algorithm.

a) Identify clustering features For building our clustering features, we focus the analysis on: (1) Subject and content reduced to references and named entities (2) Time (3) Correspondent entities. In fact, we believe that emails belonging to the same process instance are likely to have close reception dates and to share the same named entities, the same references and the same correspondents entities (from, dest, and CC).

A named entity is a real-world object, such as persons, locations, organizations, products. . . etc that can be denoted with a proper name. The references are the information generated by business applications used for executing some process tasks (e.g. customer number, purchase request ID etc.,). The purpose of reducing the email into references and named entities is to conserve only the significant contextual data in relation with the business process instances. In fact, the entire content of these email parts often contain additional vocabulary, which degrades instance detection accuracy.

The named entities are detected through two methods which are complementary in our case: We first use the Polygot NER method [2]. Named-entity recognition (NER) is a subtask of information extraction. It seeks to locate and classify named entity mentions in unstructured text into pre-defined categories such as the person names, organization, time expression, monetary values, percentage etc.Unlike the other existing pipelines (NLTK, standford, OpenNLP) where most languages are unsupported, Polygot NER is a multilingual named entity recognition tool that supports 40 major languages. It is also automatic but not complete (some named entities are not detected). To overcome this limitation, we express explicitly the non-detected patterns. As for reference detection, we inject specific regular expressions.

b) Similarity function Our similarity function is a distance function which is defined as a weighted sum of sub-distances related to our clustering features. It has the following formula:

$$f(x, y) = w_0 DC(x, y) + w_1 DT(x, y) + w_2 DNE(x, y) \tag{3}$$

Where the weights w0, w1 and w2 are defined empirically according to each process type and DC(x,y), DT(x,y) and DNE(x,y) are defined as follows:
- **DC(x, y)** is the correspondent distance between two emails x and y. We define it as a Jaccard distance between the correspondent entity sets of their interlocutors which is equal to the cardinality of their intersection divided by the cardinality of their union. More formally, let C(x) be the list of correspondents of the email x, and C(y) be the list of correspondents of the email y. DC(x,y) is then defined as follows:

$$DC(x, y) = \frac{|C(x) \cap C(y)|}{|C(x) \cup C(y)|} \tag{4}$$

- **DT(x, y)** is the time distance between emails x and y. The emails belonging to the same process instance are likely to have close reception dates. We assume that the inter-arrival duration follows an exponential law and is expressed in number of days. Consequently, we define the distance through the following formula:

$$DT(x,y) = 1 - e^{-\lambda(ts(x)-ts(y))} \tag{5}$$

ts(x) and ts(y) refer to the timestamp of x and y and λ is the time, expressed in the number of days, which separates two emails arrivals. We estimate this value by:

$$\lambda = \frac{date\_max - date\_min}{number\_of\_emails} \tag{6}$$

- **DNE(x,y)** is the distance related to the named entities and the references present in the subject and the content of the email x and those present in email y. Emails belonging to the same process instance are likely to share the same named entities and references. We define the distance DNE as a Jaccard distance:

$$DNE(x,y) = \frac{|NE(x) \cap NE(y)|}{|NE(x) \cup NE(y)|} \tag{7}$$

NE(x) is the set of named entities and references present in email x, and NE(y) is the set of named entities and references present in email y.

3.3 Step 3: Event Logs Generation

Algorithm 2 Event logs generation

```
1: procedure GenerateEventLog($pEL$)
                                                              ▷
     ▷ pEL: The pseudo event log
2:     $pEL\_ordered \leftarrow TimeBasedOrdering(pEL)$
3:     $EM\_P \leftarrow PerProcessSplitting(pEL\_ordered)$
4:     $EM\_P\_I \leftarrow InstanceDetection(EM\_P)$
5:     $InstanceIdentifierGeneration(EM\_P\_I)$
6:     $EventLogs = PerProcessGrouping(EM\_P\_I)$
7:     return $EventLogs$
```

Once the process and activity names are identified and the emails belonging to different process instances are grouped, a dataset labelled with process, activity, timestamp and instance_ID labels can be obtained. The event logs generation function aims to construct, from this new dataset, time-ordered per-process event sets. Algorithm 2 summarizes our steps to generate it. From the pseudo event logs obtained from step 1, we generate a time-ordered event logs using the email timestamp variable (line 2). Then, we split it into several subsets, each one containing a single process emails list (line 3). For each process subset, we apply a clustering algorithm in order to detect instance groups and we associate to each group a unique identifier (line 4, 5). Finally, all these groups are regrouped into a per process dataset (line 6) so that a process mining algorithm could be applied.

4 Validation

We validate our contribution through a proof of concept and experimentations carried on our dataset composed of 1026 emails and on the email environment of two employees (Microsoft Outlook as an email client, and Microsoft Exchange as an email server). In these experimentations we succeed to discover two processes: (1) a hiring process and (2) a patent application process. In this section we detail only the hiring process discovery.

4.1 Proof Of Concept

Our tool is implemented through three components:

a) Frontend component: This component is a Microsoft Outlook 2010 plugin developed using C# programming language. It has four main functionalities. First, it enables the user to manually annotate his emails. Thus, it provides the GUI that enables the user to select the email and the related process and activity names. This association is sent to the backend (SetTag interface) as a JSON object containing the email parameters and the associated process and activity names. Second, it captures incoming and outgoing messages, constructs a JSON object for each email, sends it to the backend for analysis (GetTag interface), retrieves the results (JSON object representing the detected process name and activity name) and associates the tags to the email. Third, it enables the display of the process and activity related to each email along with the email. Finally, the plugin provides users with advanced functionalities such as email search by related business process and activity.

b) Backend component: The backend component is implemented through three HTTP interfaces (SetTag, GetTag, and GetAllProcesses), with a MySQL database containing two tables: training dataset table and event logs table. The training dataset contains the following columns (id, source, destination, cc, subject, content, received_date, process_name, activity_name). The event log dataset contains the following columns (id, source, destination, cc, subject, content, received_date, process_name, activity_name, instance_id).

- SetTag Interface: This interface is used to enrich the learning database from one hand. It is invoked by the frontend when the user annotates manually an email. It receives the JSON object representing the email and the associated process and activity names. This information is inserted into the training dataset table as well as into the event log table, in which we set the instance_id column to NULL value, as we don't know yet to which instance the email belongs to.

- GetTag Interface: This interface is invoked by the Microsoft Outlook Plugin each time the user sends or receives an email. It receives a JSON object representing an email, analyzes it using the trained predictive models, and returns as a result the predicted process and activity name IDs. The result is also inserted into the event log table.
To build the machine learning models, a multinomial version of the logistic regression (LR) classifier is employed. It estimates the parameters of a logistic model for multiclass prediction task. The stochastic Gradient Descent (SGD) is used as an optimizer in the training phase. In fact, it can converge faster than batch training because it performs updates more frequently. Therefore, it has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing fields.
- GetProcesses Interface: This interface enables the email client user to display existing tags (process names and activity names). It supports him in the process of enriching the learning database. Basically, this interface is invoked when the user is about to manually annotate an email. It enables to retrieve the list of available annotation in the training dataset table. For each process name, we also retrieve the list of associated activities.

c) Instance separation component: To detect instances, we applied the clustering algorithm HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise). It extends DBSCAN by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based on the stability of clusters. The choice of HDBSCAN is justified by two reasons: (1) HDBSCAN does not require human intervention. In fact, it is a fast and non-parametric algorithm that does not require setting any parameters even the number of clusters. (2) HDBSCAN can generate clusters of different sizes, shapes and densities, which can enhance clustering accuracy.

4.2 Experimentations

We carried here experimentations to justify our choices in each step and then to evaluate their performances and to present the obtained results.

a) Validation of process and activity prediction: We manually annotated 1026 emails to obtain a correctly annotated email corpus with process and activity IDs. There are 13 processes (e.g. Hiring, PatentApplication, Command, ConferenceParticipation, travel expense refund, etc.) and 116 activities. Taking the example of hiring process, we identified the hiring steps achieved through emails: "describe the position", "publish the position", "receive applications", "setting the interviews", "asking for documents", and "notifying the decision".

The performances of process and activity prediction phase highly depend on the choice of the machine learning algorithms and the data preprocessing actions. To select these latter, we have tested different techniques until good prediction performances are reached. Better performances are noticed when:
- Using only the subject and the correspondent entities for process prediction.
- Applying the preprocessing steps detailed in II.B.2.b when we consider that the most frequent terms in the documents have a frequency greater than 5% and

the less frequent ones have a frequency less than 0.1%.
- Assuming that short emails contain less than 40 words and that email exchange history is constructed from the four previous emails.
- Setting the weights of Eq (2) as follows: $\beta_1 = 0.8$ and $\beta_2 = 0.2$.
- Using the LR with SGD optimizer for training predictive models. This result is obtained after testing two other prediction algorithms (Random Forest (RF) and Support Vector Machine (SVM)). The performances of each one were estimated by using 5-fold cross-validation method and by calculating the F1 Score. This score is a measure that combines precision and recall. Precision is known as positive predictive value while recall is called the sensitivity of the classifier. Mathematically, the F1 score is defined as:

$$F1Score = \frac{2 \times precision \times recall}{precision + recall} \tag{8}$$

The obtained F1 scores are summarized as follows: 0.8072 for Random Forest, 0.8626 for LR with SGD and 0.8094 for SVM.

b) Validation of Process Instance Detection: In this subsection, we evaluate the performances of our selected clustering technique HDBSCAN on our data set composed of 180 emails related to a hiring process.

We manually generated the emails clusters related to the process instances where we obtained 11 clusters. To compare this manual clustering with the results of HDBSCAN, we computed the Adjusted Mutual Information[4] (AMI) metric [22]. We tested different values of the weights related to distance matrix computation (Eq (3)), and we obtained an optimal configuration (using these values $w_0 = \frac{1}{2}$, $w_1 = \frac{1}{4}$, and $w_2 = \frac{1}{4}$). The AMI value obtained with this configuration is 0.86.

c) Validation of Process Model Discovery: S05cm To validate this step,

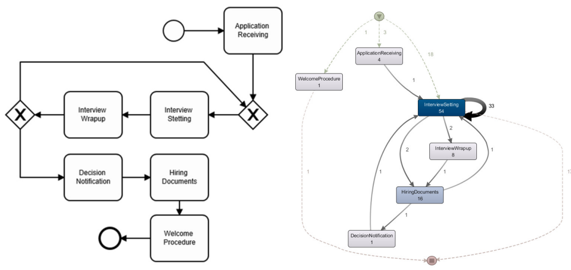

Fig. 2. Hiring process models: Real model VS Captured model

we applied the heuristic miner algorithm [23] on the automatically generated

[4] AMI computes the metric to evaluate the similarity between two hdbscan partition and real partition. It returns a value between 0 and 1. This value is closed to 1 when the two partitions are strongly matched and closed to 0 when the two partitions are weakly matched

event log of the hiring process. Figure 2 shows the theoretical and the detected model. We can notice that the behavior captured in the event log is almost in conformity with the theoretical one. Nevertheless, two discrepancy types can be detected at low frequency: (1) Unfitting model behavior which refers to behaviors observed in the theoretical model that are not allowed by the captured one (e.g.“Welcome procedure” is performed after “Decision Notification”). (2) Additional model behavior which refers to behaviors allowed in the captured model but does not exist in the theoretical one such as: “Welcome Procedure” is done initially and asking for“Hiring Documents” or “Decision Notifying” are done before “Interview Setting”. Actually, these discrepancy types can be caused either by errors accumulated through our log building system, or by the log miner technique that we have used or by a real difference between the process as observed in the emails and the related theoretical BP.

5 Related Works

Up until recently, only few studies have considered business process mining from email exchange. They have been mainly interested in: Activity and process names recognition, process instances detection and process discovery. They do require human intervention and may generate unreliable business process models. In this section, we discuss them according to three categories:

5.1 Non-learning based methods

One of the first proposals for mining business processes from emails assumes that the associated business process is explicitly included in the email subject [1]. Such an approach requires a significant human intervention and involvement. Indeed, email interlocutors must include the business process name and related attributes in the email subject, which is not realistic.

Another proposal [11] assumes that the manual task is an association of (1) classical manual task of the BPMN2.0 specification [13] and (2) a set of semantic patterns that enable to validate whether a given communication content is part of a business process, activity and a given business process instance. The limitation of such system is the necessity of anticipating and manually defining all semantic patterns related to each task which is time consuming and not scalable.

E-Mail Mining [20] is a method for semi-automatic discovery of knowledge-intensive process. From a set of emails belonging to a BP, e-Mail Mining aims to discover the amount of knowledge embedded in the execution of its activities. This knowledge consists of : (1) BP participants and their social interactions (2) Relevant terms that are related to the BP domain (3) BP activities defined by three elements : Actors, candidate actions and parameters. Relevant BP activities are selected manually from a list of candidate activities. These latter are generated after splitting emails into sentences and by assuming that each sentence is composed of : (1) a noun phrase object which can describes the agent performing an action or the resource that receives the effect of the executed action (2) a verb phrase that describes the activity performed by the agent. This approach has an interesting contribution in the field of process mining from emails. In fact, it allows the detection of multiple activities in one email as well as the metadata embedded in the execution of a given BP. However, it requires manual tasks during its execution (e.g for selecting relevant activities or sample of emails related to one BP). Moreover, it considers emails as storytelling textual data to mine the candidate activities. Actually , emails do not have the same structure as narrative textual data (which generally describes activities in a more formal way than e-mails). For instance, the proposal does not seem to handle passive-voice sentences where actors do not appear or where their positions are switched with those of resources.

5.2 Act theory based methods

This category deals with activity name recognition by using act theory based methods: The idea behind this theory is to classify emails according to the sender's intent [19,18]. Thus, two possible classifications of speech acts are proposed: (1) Illocutionary act classes; Assertive, Commissive, Directive, Expressive, declarations. (2) Speech act verbs: Propose, Request, Deliver, Commit, etc. Some proposals set email speech acts in advance. Then, they apply a supervised learning algorithm to classify emails as containing or not containing the specific acts [4,17]. Other works treat the problem of process detection as a problem of conversation finder such as [12]: It suggests firstly classifying emails into business and non-business process related. The business-oriented email messages are then grouped into threads to detect conversations using a refined version of Vector Space Model and a semantic similarity measure. Finally, the interactions in each conversation are labeled by applying the classification of illocutionary acts [19].

An iterative relational learning approach for email task management was also suggested [10]. It exploits the mutual performance improvement between the extraction of speech acts and the identification of related emails. In fact, after initializing both of them using automatic methods, a supervised learning algorithm (SMO implementation of SVM [16]) is applied on incoming emails: It takes into account related emails as a feature to recognize the correspondent speech acts. Then, a relational learning terminology [14] is exploited: It similarly uses speech act as a feature to predict relations between the incoming and the existing messages.

Obviously, all of these works require labeled data for training statistical speech acts recognizers which leads to a huge human intervention. Furthermore, business process tasks differ from one process to another. Thus, setting a unique list of activities in advance, degrades the performances of generating the right business process models.

5.3 Unsupervised learning based methods

In order to minimize the human intervention and to avoid preparing a labeled dataset, there exist propositions that have integrated unsupervised learning techniques in their approaches to mine business processes from emails [8,9,6,5]. One of these propositions identify the process and the instance clusters by applying a hierarchical clustering method (Bottom up) [8]. In order to find process groups, the distance used combines the subject and body attributes. Then, to detect instances, the timestamp attribute is added. As for the process activities identification phase, the K-mean algorithm is adopted. The approach proposes a customization method to set the number (K) and the initial centers, on the basis of the instance clusters obtained from the previous step. Then, it applies a distance formula that takes into consideration the meanings similarity of the words present in the subjects and the bodies.

Another approach proposes a two step algorithm to discover the processes and activities from emails [9]. A hierarchical clustering is sequentially applied: first to deduce process clusters and then to deduce sub-clusters corresponding to activity types. The similarity measurement is based on word2vect method: It aims to exploit the hidden semantic relations between words existing in email contents and subjects. An activity labeling technique is also proposed: It recommends to the user the most frequent contiguous sequence of n items existing in an activity type cluster.

These studies aim to minimize the human intervention. However, they have some limitations: First, the hierarchical clustering requires a human effort in tuning its parameters. Additionally, its quality highly depends on how these parameters are set [15,3]. Second, it is computational hard. Hence, applying the same algorithm twice in the same method increases the computational complexity and the execution time [8]. Third, the activity identification quality highly depends on the instance clustering phase [8]. In fact, as the K-mean algorithm is sensitive to the initial start centers, poor instance detection quality can certainly lead to a bad convergence. Finally, the automatic generation of labels considerably increases the risk of error and interpretation [9].

MailOfMine [6,5] proposes to mine artful business processes and to define them with a "declarative" approach. It suggests to start from some assumptions which map email and BP structures: (1) Each conversation presents an activity trace (2) Each activity presents a set of elementary tasks deduced from conversation key parts (3) Each process is composed by a set of activities. MailOfMine approach consists basically of: (1) Applying three times a similarity clustering algorithm: to cluster emails into conversation threads, to cluster these threads into activity types and finally, to cluster each activity key parts into task types. During the clustering process, email body, the names of attached files and some email header fields are taken into consideration.(3) Applying supervised learning process to assign activities to different processes (4) Automatically labeling activity tasks with the possibility of customizing them and manually assigning activity and process names (5) Mining constraints between tasks (activities) among each activity (process) . The proposed work has the advantage of discovering BP with different level of granularity (Process, subprocess or activity, task) and describing them with declarative approach, which is more flexible than the classical imperative approach. Nevertheless, this work suffers from some limitations: (1) Its execution time can diverge when applying it on large number of traces containing various tasks and activity types, e.g; it is linear (quadratic) time with respect to the number, size of traces. (2) A considerable human intervention is required to manually assigning activity and process names and to initiate activity classification step.

6 Conclusion & Discussion

In this paper, a solution for mining business processes from email logs was proposed and evaluated through a proof of concept implementation and successful experimentations on our dataset.

To build a training dataset from existing corpus of emails, we proposed a collaborative and iterative approach which is implemented through graphical interfaces and automatic prediction functionalities. This has the advantage of encouraging users to be involved in building an annotated dataset since it facilitates the tagging task and minimizes the required effort. Consequently, the training dataset will be built gradually and will be available instantly without the need to dispose a lot of time and human resources. However, this approach still relies on human involvement. Moreover, tagging collaboratively the same dataset can lead to dispose samples belonging to the same cluster but with different annotations. Therefore, tag normalization step is required.

The prediction entity is based on a supervised learning technique. For building classification features, we have investigated, according to the prediction goal, some or all of these variables: email participant correspondent entities, subject, content and exchange history. Our experiments revealed that email contents degrade the performances of process name recognition. Even if this assumption seems contradictory to existing works [8,9], it can be justified by the nature of our dataset and our preprocessing steps.

Our instance detection approach differs from related works by using a fast and non parametric clustering algorithm which is non sensitive to the noise and which can generate clusters with different shapes, sizes and densities. Moreover, we have reduced the body and the subject into references and named entities for clustering emails into BP instances. To improve the detection quality of named entities and references, we have defined explicitly the non-detected patterns. This action has the advantage of having good performances, however, it is manual and by consequence costly. We have assumed that three variables (timestamp, correspondent entities and named entities) can contribute to separate BP instances. Actually, the contribution of each variable depends on the nature of the BP. This is why we have introduced weights correlated to each variable to be tuned by users according to their expertise. For instance, the timestamp variable can help to separate instances in the case of BP with time constraints (e.g; the accounting closing process that is carried out regularly on scheduled dates) while in the case of BP whose instances are independent of time, the same variable seems to have no effect.

In our approach, we have handled some research questions related to BP mining from emails by supposing that one email can be affected to one process, one activity and one instance. Actually, messaging systems such as emails allows users to discuss BP issues with informal way without respecting such constraints; in one email, user can discuss more than one activity and more than one instance. This kind of challenges was addressed in previous works such as [20] by assuming that activities are expressed and generally correlated with their metadata at sentence level. In future works, we plan to study these challenges. We plan also to more automate the BP discovery pipeline since the current approach still requires human involvement. Finally, we suggest to employ similarity meaning measures for constructing learning features based on email contents.

References

1. van der Aalst, W.M., Nikolov, A.: Emailanalyzer: an e-mail mining plug-in for the prom framework. BPM Center Report BPM-07-16, BPMCenter. org (2007)
2. Al-Rfou, R., et al.: Polyglot-ner: Massive multilingual named entity recognition. In: SIAM International Conference on Data Mining. pp. 586–594. SIAM (2015)
3. Ciosici, M.R.: Improving quality of hierarchical clustering for large data series. arXiv preprint arXiv:1608.01238 (2016)
4. Cohen, W.W., et al.: Learning to classify email into"speech acts". In: Empirical Methods in Natural Language Processing (2004)
5. Di Ciccio, C., Mecella, M.: Minerful, a mining algorithm for declarative process constraints in mailofmine. Department of Computer and System Sciences Antonio Ruberti Technical Reports **4**(3) (2012)
6. Di Ciccio, C., Mecella, M., Scannapieco, M., Zardetto, D., Catarci, T.: Mailofmine–analyzing mail messages for mining artful collaborative processes. In: International Symposium on Data-Driven Process Discovery and Analysis. pp. 55–81. Springer (2011)
7. Jeong, M., et al.: Semi-supervised speech act recognition in emails and forums. In: Empirical Methods in Natural Language Processing. vol. 3, pp. 1250–1259. Association for Computational Linguistics (2009)
8. Jlailaty, D., et al.: A framework for mining process models from emails logs. arXiv preprint arXiv:1609.06127 (2016)
9. Jlailaty, D., et al.: Mining business process activities from email logs. In: Cognitive Computing (ICCC). pp. 112–119. IEEE (2017)
10. Khoussainov, R., Kushmerick, N.: Email task management: An iterative relational learning approach. In: CEAS (2005)
11. Laga, N., et al.: Communication-based business process task detection-application in the crm context. In: Enterprise Distributed Object Computing Workshop (EDOCW). pp. 1–8. IEEE (2016)
12. Mavaddat, M., et al.: Facilitating business process discovery using email analysis. In: The First International Conference on Business Intelligence and Technology. Citeseer (2011)
13. Model, B.P.: Notation (bpmn) version 2.0. OMG Specification, Object Management Group pp. 22–31 (2011)

14. Neville, J., Jensen, D.: Iterative classification in relational data. In: Learning Statistical Models from Relational Data. pp. 13–20 (2000)
15. Oyang, Y.J., et al.: Characteristics of a hierarchical data clustering algorithm based on gravity theory. Tech. rep., Technical Report of NTUCSIE 02-01.(Available at http://mars. csie. ntu. edu . . . (2001)
16. Platt, J.C.: 12 fast training of support vector machines using sequential minimal optimization. Advances in kernel methods pp. 185–208 (1999)
17. Qadir, A., Riloff, E.: Classifying sentences as speech acts in message board posts. In: Empirical Methods in Natural Language Processing. pp. 748–758. Association for Computational Linguistics (2011)
18. Searle, J.R.: A taxonomy of illocutionary acts (1975)
19. Searle, J.R., Searle, J.R.: Speech acts: An essay in the philosophy of language, vol. 626. Cambridge university press (1969)
20. Soares, D.C., Santoro, F.M., Baião, F.A.: Discovering collaborative knowledge-intensive processes through e-mail mining. Journal of Network and Computer Applications **36**(6), 1451–1465 (2013)
21. Van Der Aalst, W., et al.: Process mining manifesto. In: International Conference on Business Process Management. pp. 169–194. Springer (2011)
22. Vinh, N., et al.: Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. Journal of Machine Learning Research **11**(Oct), 2837–2854 (2010)
23. Weijters, A., et al.: Process mining with the heuristics miner-algorithm. Technische Universiteit Eindhoven, Tech. Rep. WP **166**, 1–34 (2006)