Towards Efficient Business Process Clustering and Retrieval: Combining Language Modeling and Structure Matching

Mu Qiao[1], Rama Akkiraju[2], and Aubrey J. Rembert[2]

[1] Department of Computer Science and Engineering,
The Pennsylvania State University, University Park, PA 16802, USA
muq103@cse.psu.edu
[2] IBM T.J. Watson Research Center, Hawthorne, NY 10532, USA
{akkiraju,ajrember}@us.ibm.com

Abstract. Large organizations tend to have hundreds of business processes. Discovering and understanding similarities among business processes can be useful to organizations for a number of reasons including better overall process management and maintenance. In this paper we present a novel and efficient approach to cluster and retrieve business processes. A given set of business processes are clustered based on their underlying topic, structure and semantic similarities. In addition, given a query business process, top k most similar processes are retrieved based on clustering results. In this work, we bring together two not well-connected schools of work: statistical language modeling and structure matching and combine them in a novel way. Our approach takes into account both high-level topic information that can be collected from process description documents and keywords as well as detailed structural features such as process control flows in finding similarities among business processes. This ability to work with processes that may not always have formal control flows is particularly useful in dealing with real-world business processes which are not always described formally. We developed a system to implement our approach and evaluated it on several collections of industry best practice processes and real-world business processes at a large IT service company that are described at varied levels of formalisms. Our experimental results reveal that the combined language modeling and structure matching based retrieval outperforms structure-matching-only techniques in both mean average precision and running time measures.

1 Introduction

Large organizations tend to have hundreds of business processes. Discovering and understanding the similarities among business processes can be useful to organizations for many reasons. First, commonly occurring process activities can be managed more efficiently resulting in business process maintenance efficiencies. Second, common and similar processes and process activities can be reused

S. Rinderle-Ma, F. Toumani, and K. Wolf (Eds.): BPM 2011, LNCS 6896, pp. 199–214, 2011.

in new or changed process implementations. Third, commonalities among processes and process activities can help identify opportunities for standardization and consolidation of business processes. For example, in the context of a merger or acquisition, discovering similar business processes can aid in decisions about how the merging organizations' processes can be effectively integrated.

The majority of the business process similarity and retrieval research community is concerned with computing similarity primarily by using only the structural component of process models. Some of these approaches rely on full structure comparison, which is computationally expensive, while others rely on local structure comparison, i.e. the successors and predecessors of an activity. More often than not, business processes currently running in companies have been developed and deployed sometime ago, and formal models rarely exist and would be hard or too expensive to re-construct formally. What might be available is informal unstructured or semi-structured documentation. Therefore, a business process clustering and retrieval system that can deal with processes that have different levels of partial information represented in varying degrees of formalism is critical to be useful in real world. We address this problem in our work.

In this paper, we present a language modeling and structure matching based business process clustering and retrieval method that takes into account both the process structural and textual information. We formally define the problem of *business process clustering and retrieval* as follows. Let $\mathbf{P}$ be a business process model repository, where each process model $P_i \in \mathbf{P}$ is the tuple (S_i, T_i) such that S_i represents structural information, e.g., control-flow and data-flow, and T_i is textual information, e.g., functionality descriptions. Given $\mathbf{P}$, a business process clustering function assigns P_i's into subsets (aka clusters) so that processes in the same subset are close to each other while those in different subsets are as distinct as possible. Also, given a query business process model P_Q, a retrieval function returns the top k process models ranked in descending relevance order, where similarity is an approximation of relevance. Our approach not only speeds up the retrieval process, but also provides an intuitive organization of the business process repository. Experiments on several collections of industry best practice processes (e.g., SAP, APQC's process classification framework processes) and real-world business processes at a large IT service company show that our retrieval method has superior mean average precision and execution times than purely structural approaches.

2 Related Work

Business process clustering and retrieval are defined with respect to process similarity measures. The most common approach to process similarity measures is computing the graph edit distance between control-flow graphs of processes. Dijkman et al. [4] compared four techniques for control-flow graph comparison: the A-star algorithm, a greedy approach, a heuristic approach, and an exhaustive approach. The A-star algorithm performed the best. Madhusudan et al. [15] combined control-flow comparison with similarity flooding. This approach is

highly sensitive, and only near exact matches pass the stringent comparison method, which is a requirement of their particular application. Graph comparison approaches give compelling precision results, however, they are not practical. Real-world processes tend to be informally described and most of the information about the process is not encoded in its control-flow graph.

Many approaches reduce the reliance on full graph comparisons, and attempt to exploit additional information. The work of Ehrig et al. [8] relies on activity labels and activity context to compute process similarity. Van Dongen et al. [6] use causal footprints to measure similarity, which capture the behavior information about processes and are defined in a vector space. A two-level process comparison approach is proposed by Yan et al. [21]. At the first level, activity labels, and local activity structure are used. The result of the first level is a categorization: relevant, potentially relevant, and irrelevant. Potentially relevant process models are compared using a full graph comparison technique.

To the best of our knowledge, no previous work has studied the clustering of real-world business processes. Some initial efforts are made in [11], where two types of vectors, i.e., activity and transition vectors, are proposed to calculate the pairwise structural distance between processes. These distances are then used to create a hierarchical clustering tree. This approach was only evaluated on a small scale synthetic data set and the proposed vectors may not be sufficient to capture the structural features of real-world processes. Business process retrieval, mostly under the name process similarity search, has been explored in [4,21,10] using graph comparison. Recent survey [7] discusses the developments of process similarity search techniques.

In general, our business process clustering and retrieval method extends the graph comparison approaches by considering text features, such as topic similarities. The process retrieval is based on the trained clustering model and clustering results. Unlike the work of Dong et al. [5], where clustering is applied to parameter names of web-service operations and then leveraged to measure the web-service similarity, we do clustering on process models themselves. Therefore, clustering not only facilitates the retrieval but also organizes processes in a meaningful way.

3 Business Process Clustering and Retrieval

The flow diagram of our proposed business process clustering and retrieval method is shown in Fig. 1. Throughout the paper we use the terms "similarities" and "commonalities", "activities" and "steps", interchangeably.

For a selected business process repository, we first extract the textual and structural features from each process model. All the text terms from step names and the documentation will form a text feature vector for each process. Note that in our process collections, the documentation of a process is a paragraph of text description, introducing its purposes and specific functionalities. The process control flow graphs are stored as structural features. We propose a two-level process clustering method. Specifically, a topic language modeling approach is

first applied to the input processes represented by text feature vectors to obtain the first level clustering based on their topic similarities. Within each cluster, detailed structure similarity match is then performed between all the processes. A clustering method based on pairwise structure similarities is employed to obtain the second level clusters. In Fig. 1, C_{ij} indicates the jth second level cluster within the ith first level cluster. Results from the first level clustering, such as the topic mixture (probabilities of topics appearing in each process), the topic word distribution (probabilities of text terms associated with each topic), and the process clusters, are kept for future retrieval usage. These high-level and granular clusters can reveal the commonalities among business processes. Once the clusters are created, a retrieval function will use the trained topic language model and clusters to retrieve similar business processes for a given query. A ranking scheme ranks the processes in descending order of relevance/similarity to the query. Details of the proposed process clustering and retrieval method are discussed in the following sections.

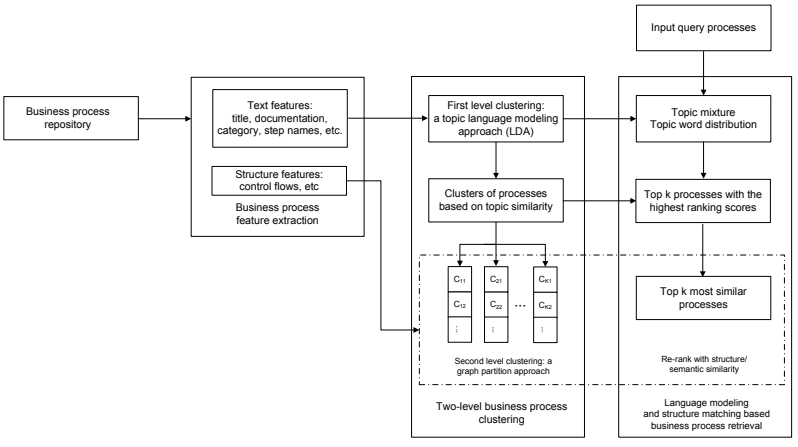

Fig. 1. The flow diagram of language modeling and structure matching based business processes clustering and retrieval method

3.1 Two-Level Business Process Clustering

Business process clustering helps reveal the underlying characteristics and commonalities among a large collection of processes. The information extracted by clustering can also facilitate subsequent analysis, for instance, to speed up text documents or images indexing and retrieval [13].

Business process models contain *compound data*. By *compound*, we mean a group of data types are needed to represent the object. For instance, process models contain both textual features which can be represented by the vector-space model [17] and structural features usually represented by graphs. In this work, we present a two-level business process clustering method which incorporates these two kinds of information. At the first level, processes are clustered based on their topic similarities. For instance, processes with the same topic as

"order" and "management" will be grouped together. Within each cluster, we further take into account the detailed structure information of processes and cluster them based on their structure similarities. For instance, at the second level, processes that are related to "order fulfillment" and "order entry" may be assigned into their own separate clusters.

There are several advantages of the two-level method. First, we not only find processes with close structures but also discover processes that describe the same topic but have completely different structures. Second, since the pairwise structure similarity matching is only conducted on processes within each first level cluster, the computation time is significantly reduced as compared to the clustering approach based on all pairwise structure similarities. Additionally, in real practice, some processes may have incomplete structure or no structure at all. We can still do clustering on those processes based on their text features.

First level clustering: a topic language modeling approach. We employ topic language modeling, specifically, Latent Dirichlet Allocation (LDA) [2], to cluster processes based on their topic similarities. LDA is used to model the generative process of a text document corpus, where a document is summarized as a mixture of topics. In our work, text terms (aka words) of each business process, e.g., from activity labels and the process documentation, are extracted and combined to form a document represented by a text feature vector. All the processes in the repository are thus converted into a collection of documents in terms of their text features. In LDA, each document is modeled as a mixture over a set of latent topics. We assume that the text document created for each business process is represented by a mixture of topics, for instance, *management, purchase, order,* etc. The topic mixture is drawn from a Dirichlet prior over the entire collection. Suppose a document d is a word vector of length N_d, denoted by $\mathbf{w} = (w_1, w_2, ..., w_{N_d})$, where w_n is the nth word in the document d. LDA is represented as a graphical model in Fig. 2. The generative process for each document in a collection is as follows:

1. For each topic z, choose a multinomial distribution ϕ_z from a Dirichlet prior with parameter β;
2. For each document d, choose a multinomial distribution θ_d over topics from a Dirichlet prior with parameter α;
3. For each of the N_d word w_i in document d, choose a topic z_i from the multinomial distribution θ_d;
4. Choose a word w from the multinomial distribution ϕ_z.

Exact inference in LDA is generally intractable. Gibbs sampling, a special case of the Markov chain Monte Carlo (MCMC) method, has been used to estimate the parameters of the LDA model, which often yields relatively simple algorithms for approximate inference [9]. Due to space limitation, we refer interested readers to [2] and [9] for the details of LDA and gibbs sampling.

In Fig. 2, the parameter θ_d is the topic distribution (or mixture) in document d and ϕ_k is the word distribution in topic k. Intuitively, we can interpret θ_d as the probabilities of topics appearing in document d and ϕ_k as the probabilities

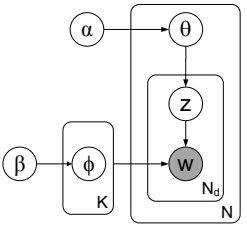

Fig. 2. Graphical model representation of LDA. K is the number of topics; N is the number of documents; N_d is the number of words in document d.

of words associated with topic k. Since each document represents the text features of a process, we finally get the topic mixture for each process. The topic possessing the largest posterior probability is chosen for that process. Processes associated with the same topic form a cluster. Moreover, the topic mixture obtained from LDA can be readily used as a soft clustering, i.e., a process can be assigned to multiple clusters with different probabilities.

Cluster number selection. The perplexity score has been widely used in LDA to determine the number of topics (aka clusters in this work), which is a standard measure to evaluate the prediction power of a probabilistic model. The perplexity of a set of words from an unseen document $d \in D_{test}$ documents, is defined as the exponential of the negative normalized predictive log-likelihood under the training model. The perplexity is monotonically decreasing in the likelihood of the test data [2]. Therefore, a lower perplexity score over a held-out document set indicates a better generalization performance. The value of k, which results in the smallest perplexity score over a randomly selected set of test documents, is selected as the cluster number.

Second level clustering: a graph partition approach. After the first level clustering, we look into the detailed structural features of the processes within each cluster. Clustering methods based on pairwise distances, such as graph partition, have enjoyed wide spread applications when a mathematical representation for the object is difficult or intractable to obtain. Since the structure of a business process model cannot be easily represented by a single data type, such as vectors, the graph partition method is applied to obtain the second level clustering results, which is illustrated by Fig. 3. Each business process is represented by a node, denoted by P_i. If the structure similarity between two processes is above a certain threshold, there is an edge between the corresponding nodes. In Fig. 3, nodes with the same color belong to the same first level cluster. All the nodes in the same connected component form a second level cluster.

In our work, different structure matching algorithms are selected according to the characteristic of specific business process repository. The detailed structure matching approaches are described in Section 4, after we introduce the business process repositories.

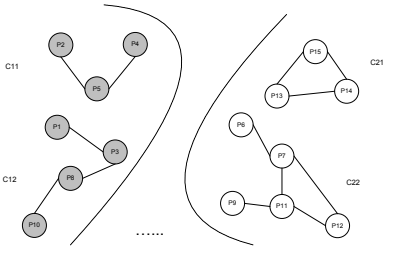

Fig. 3. Graph partition to find the second level clusters

3.2 Language Modeling and Structure Matching Based Business Process Retrieval

In our method, language modeling and structure matching are combined for process retrieval. Given a query process, processes are first ranked based on their text features using a language modeling approach. Since the important text features of the query process and each process in the repository are represented as text documents, we essentially retrieve a subset of documents which are most similar to the query document. A LDA-based retrieval method is employed, which has improved documents retrieval results within a language model framework [20].

The basic idea of using language model based retrieval is to model the query generation process [16]. A language model D is built for each document in the collection and the documents are ranked based on their probabilities of generating the query Q, i.e., $P(Q|D)$. Assuming the words in the query Q are all independent, we have $P(Q|D) = \prod_{i=1}^{m} P(q_i|D)$, where q_i is the ith word in the query Q and m is the total number of words. $P(q_i|D)$ is specified by

$$P(q_i|D) = \lambda P_{ml}(q_i|D) + (1-\lambda)P(q_i|Coll)\ , \tag{1}$$

where $P_{ml}(q_i|D)$ and $P(q_i|Coll)$ are the maximum likelihood estimates of word q_i in the document model D and in the entire collection. This model is called document based retrieval. In (1), λ is a general smoothing term, which has different forms for different smoothing methods. A study of different smoothing methods for language models applied to retrieval can be found in [22]. For instance, λ takes the form $N_d/(N_d + \mu)$ for Dirichlet smoothing, where μ is a smoothing parameter.

A LDA-based model is proposed in [20] for documents retrieval within a language model framework. Experimental results showed that it outperforms the cluster based retrieval method [14] which incorporates the clustering information as Dirichlet smoothing. Specifically, $P(q_i|D)$ is now modeled as a linear combination of the original document based retrieval model (1) and the LDA model:

$$P(q_i|D) = (1-\gamma)(\frac{N_d}{N_d+\mu}P_{ml}(q_i|D) + (1-\frac{N_d}{N_d+\mu})P(q_i|Coll)) + \gamma P_{lda}(q_i|D)\ , \tag{2}$$

where $N_d/(N_d+\mu)$ is the Dirichlet smoothing term, $P_{lda}(q_i|D)$ is the probability that q_i is generated by D in the LDA model and γ is a weight term. $P_{lda}(q_i|D)$ can be directly obtained from the trained LDA model in the process clustering step, i.e., $P_{lda}(q_i|D) = \sum_{z=1}^{K} P(q_i|D,\hat{\phi})P(z|\hat{\theta},D)$, where $\hat{\theta}$, $\hat{\phi}$ are the posterior estimates of θ and ϕ in LDA.

We then calculate $P(Q|D)$ for the language model D built for each document d, which is used to rank the process corresponding to that document. A larger value indicates that the process has a higher probability to generate the query, i.e., closer to the query in terms of their text similarities. In this work, we adopt the aforementioned LDA-based retrieval method to first rank all the processes based on their text features and obtain a list of top k processes. In the second step, detailed structure similarities between the query and the processes in the list are computed. These processes are then re-ranked by their structure similarities in a descending order. Let r_i, t_i and s_i denote the rank, text similarity, and structure similarity of process i in the list, respectively. The ranking scheme can be expressed as follows:

$$r_i \prec r_j \; if \; \begin{cases} s_i > s_j \\ t_i > t_j, \; s_i = s_j \end{cases},$$

which incorporates the perspectives from both textual and structural information. We name this process retrieval method as *LDA-retrieval*. Since we only need to compute the structure similarities between the query process and a small subset of processes (i.e., k processes), the computational complexity is significantly reduced. In addition, if the query process or some process in the repository does not have formal control flows, we can still retrieve processes which match the user's query interest based on their text similarities.

For comparison, we propose another cluster-based process retrieval method. Specifically, it first determines which topic (aka cluster) the query process falls under, and then does structural comparisons between the query and processes in that cluster. Finally, processes with the top k highest structure similarity scores are retrieved. Clustering therefore serves as a filtering tool. Some early studies have shown that this type of retrieval has advantages on precision-oriented searches for text documents [3]. However, other experimental work also indicated that clustering may not necessarily improve the retrieval performance, depending on specific domains or the collections [19]. Note that the clusters employed in this retrieval method are from the first level clustering results in the process clustering step. The reason that we do not use the second level clusters is that the cluster membership of an unseen query process cannot be directly determined on the second level. Since these clusters are also obtained by LDA, we name this process retrieval method as *LDA-cluster-retrieval*. Similarly, the clusters can also be obtained by applying the well know kmeans algorithm to processes represented by text feature vectors. As comparisons, we name the retrieval method based on kmeans as *kmeans-cluster-retrieval*.

As a summary, we implemented and compared four business process retrieval systems, i.e., *structure-matching-retrieval*, *LDA-retrieval*, *LDA-cluster-retrieval*, and *kmeans-cluster-retrieval*. The trained LDA model and clusters from the

business process clustering step are reused in *LDA-retrieval* and *LDA-cluster-retrieval*. Detailed evaluations of these methods are presented in Sect. 4.

4 Experimental Results

In this section, we present the experimental results of the proposed business process clustering and retrieval method on multiple business process repositories.

Given a business process repository, we first extract text words from step names and the documentation (if available) of each process and combine them to form a text document. After removing a standard list of stop words, we get a text corpus for the entire process repository. If the associated text document for a process is too short, the process will be excluded. We select processes whose number of text terms is no less than a certain threshold, which is set to 50, unless otherwise noted. Below we explain each data set briefly.

SAP best practice processes: we extract around six hundred business processes from SAP's solution composer tool which is publicly available[1]. There are two categories of processes, i.e., industry-specific and cross-industry. 327 processes from industry-specific domain are obtained. The total number of process steps in these processes is 3291, among which 2425 are unique. Similarly, we obtain 276 cross-industry processes which have 2752 steps in total and 2012 unique steps. The structure information of all these processes is not available.

SAP reference models: this is a collection of 604 business process models supported by the SAP enterprise system [12]. They are represented in the event driven process chains (EPC) notation, with moderately complicated structures. Since processes do not have documentations, we extract the text terms from step names to form a document for each process. After weeding out processes with fewer terms than the set threshold, we get 242 processes. The total number of steps of these processes is 6302, among which 2982 are unique.

APQC process classification framework: we obtain the industry-specific processes from American Product Quality Council's (APQC) process classification framework (PCF)[2]. These processes do not have documentations. The text terms are extracted from step names. Since the average number of terms of these processes are smaller than those in previous data sets, a smaller threshold of 20 is set to remove processes with short text. We finally obtain 180 processes. The total number of steps for these processes is 1312, among which 1269 are unique. The structures of these processes are linear chains.

Large company real-world processes: this data set contains a collection of real world processes at a large IT service company. All the processes fall roughly into four categories: finance (FIN), master data management (MDM), order to cash (O2C), and opportunity to order (O2O). After removing process with short

[1] http://www.sap.com/solutions/businessmaps/composer/index.epx
[2] http://www.apqc.org/

text, we finally get 117 processes. The total number of steps in these processes is 2220, among which 1862 are unique.

In our business process clustering and retrieval method, structure similarity matching is used to compute the pairwise similarities between processes within each first level cluster and also the similarities between the query process and a selected small subset of processes for retrieval. Different structure matching algorithms are selected according to the characteristic of specific business process repository. We consider the process step names in structure matching[3]. More process attributes, for instance, data flow and textual documentation for specific process steps, can also be taken into account. However, how to combine attributes from different modularities to achieve better structure comparison is not a trivial question. Since it is not the focus of this paper, we will not discuss it in great depth.

For the SAP reference model repository, the A-star graph matching algorithm is used since it has good performance in terms of mean average precision and running time shown in previous work [4]. In each step, the algorithm selects the existing partial mapping with the largest graph edit similarity, yielding an optimal mapping in the end. For processes in APQC process classification framework, since they have the chain structures and contain standardized steps, we adopt the "string edit" distance to measure their structure similarities, where each process is modeled as a "string" and the process steps are "characters". The SAP best practice processes only have process steps without specifying the control flows. A semantic similarity score [1] is therefore defined to measure the similarity between processes, i.e., $S = 2 \times n_{cs}/(n_p + n_q)$, where n_p and n_q are the numbers of steps in process p and q, and n_{cs} is the number of common steps. Also, the structures of our evaluated real-world processes from a large IT service company are very complicated and they rarely share any commonalities. For this repository, we also use the aforementioned semantic similarity scores to compute their similarities.

4.1 Business Process Clustering Evaluation

We present in this section the two-level business process clustering results. The Dirichlet priors α and β of LDA in the first level clustering are set to 0.2 and 0.1, which are common settings in literature. The number of iterations of the Markov chain for gibbs sampling is set to 3000. The larger the number of iterations, the longer time it takes to train the model. For the data sets described above, we note that gibbs sampling usually converges before 3,000 iterations. Before doing the first level clustering with LDA, we first determine the number of topics (aka clusters) using the perplexity score introduced in Sect. 3.1. The LDA model is trained separately on each data set. We calculate the perplexity scores of the trained model over a held-out test data set with different number of topics. Specifically, 10% samples from each data set are randomly selected as the test

[3] For SAP reference models represented by EPC, the process steps considered in structure matching consist of both functions and events.

set. For each data set, the number of topics is set to the value resulting in the smallest perplexity score. Due to space limitation, we will not show how the perplexity score changes when the number of topics varies.

Discovered Topics. We examine the top words associated with each topic to evaluate the quality of discovered topics. Table 1 shows a subset of the discovered topics from two process repositories. For instance, in Table 1(a), the four discovered topics in SAP best practice (industry) repository are "contract", "management", "planning", and "payment". If we look at their associated top words, most of them are related with that particular topic. Note that the word in "discovered topics" is the key word we select to represent the topic. Business processes associated with the same topic will form a cluster on the first level of the clustering.

Table 1. Top words associated with the selected topics

(a) SAP best practice (industry)

id	discovered topics	top words
0	contract	contract, system, billing, contracts, settlement, data, accounting, rights
1	management	level, process, control, management, controls, data, define, assign, perform
3	planning	planning, orders, project, order, production, process, planned, mrp, plan
4	payment	system, account, payment, business, invoice, credit, check, document, create

(b) SAP reference models

id	discovered topics	top words
0	asset	asset, depreciation, posted, investment, master, fixed, changed, simulation
2	maintenance	order, maintenance, material, created, capacity, service, purchase, deleted
3	quality	inspection, quality, lot, created, defects, certificate, vendor, results
14	registration	business, event, exists, order, appraisal, cancelled, resource, attendance

Two-level Clustering Results. After processes are clustered initially based on their topic similarities, we will do a detailed structure similarity matching among all the processes within the same cluster. Note that, for SAP reference models, we use the process heuristic matching algorithm from [4] with default setting, rather than the A-star algorithm, due to the extremely long running time we have observed using A-star for this particular problem. Graph partition method is then employed to generate the second level clusters. In practice, if the structure similarity score between two processes is above 0.0, there will be an edge between the corresponding nodes. Finally, nodes in the same connected component in the graph form a cluster.

Table 2 shows selected two-level clustering results on several process repositories. All the processes in the same table belong to the same first level cluster. For instance, in Table 2(a), most processes are about planning, although they may discuss different kinds of planning, e.g., production planning, supply network planning, distribution planning and relocation planning. The last column shows the probabilities of these processes belonging to this specific cluster. Processes having close structure similarities are further grouped together, indicated by the same cluster id in the first column of each table. C_{ij} indicates the jth second level cluster within the ith first level cluster. In Table 2(a), processes in cluster C_{31} are about production planning in ERP, while those in cluster C_{32} are about production planning in SCM, though all of them are about production planning. Processes that do not have any common structure with others

will be stand-alone, indicated by C_{in} in their cluster id, where i is the first level cluster id. Results in Table 2(b) are based on the SAP reference models. All the processes in this cluster are related with "maintenance and service", which are from three categories, i.e., "plant maintenance", "customer service", and "quality management". Looking into the detailed steps of these processes, we found that most of them are about maintenance, service, and material order, sharing partial structures, though they are from different categories. Since the process title information of this data set is not available, we put their process labels in the title column instead. Some selected clustering results of real-world business processes from a large IT services company are shown in Table 2(c). Most processes in this cluster are related with "order", although they may be from different categories. On the second level clustering, processes from the same category tend to be clustered together since there is a better chance that processes in the same category have similar structures.

Table 2. Selected Two-level Clustering Results

(a) Topic 3 (planning) in SAP best practice (industry)

cls id	category	process title	prob.
C_{31}	Mining	Production Planning (Process Manufacturing) MTS in ERP	0.94
C_{31}	Mining	Production Planning (Process Manufacturing) MTO in ERP	0.97
C_{31}	Aero. Def. Manufacturing	Production Planning (Discrete Manufacturing) MTO in ERP	0.97
C_{31}	Aero. Def. Manufacturing	Production Planning (Discrete Manufacturing) CTO in ERP	0.96
C_{31}	Aero. Def. Manufacturing	Production Planning (Discrete Manufacturing) MTS in ERP	0.93
C_{32}	Mining	Production Planning (Process Manufacturing) MTO in SCM	0.97
C_{32}	Mining	Production Planning (Process Manufacturing) MTS in SCM	0.93
C_{32}	Consumer Products-Food	Production Planning (Repetitive Manufacturing) CTO in SCM	0.91
C_{32}	Aero. Def. Manufacturing	Production Planning (Discrete Manufacturing) CTO in SCM	0.96
C_{33}	Mining	Supply Network Planning Heuristic	0.95
C_{33}	Mining	Supply Network Optimization	0.95
C_{33}	Mining	Multilevel Demand and Supply Match	0.97
C_{3n}	Pharmaceuticals	Distribution Planning	0.73
C_{3n}	Defense	Relocation Planning	0.64
C_{3n}	Aero. Def. Manufacturing	Pre-Planning of Long Lead-Time Items	0.62

(b) Topic 2 (maintenance and service) in SAP reference models

cls id	category	process title	prob.
C_{21}	Customer Service	1Ku_a1xs	0.94
C_{21}	Customer Service	1Ku_9efq	0.94
C_{21}	Plant Maintenance	1In_at4y	0.97
C_{21}	Plant Maintenance	1In_bip2	0.98
C_{21}	Plant Maintenance	1In_b8et	0.97
C_{21}	Plant Maintenance	1In_apbf	0.96
C_{21}	Quality Management	1Qu_c117	0.84
C_{21}	Quality Management	1Qu_c3ie	0.86
C_{22}	Customer Service	1Ku_903f	0.35
C_{22}	Customer Service	1Ku_9mgu	0.40
C_{22}	Plant Maintenance	1In_bb6y	0.50
C_{22}	Plant Maintenance	1In_agyu	0.65
C_{22}	Plant Maintenance	1In_b19m	0.69
C_{23}	Customer Service	1Ku_9rnu	0.93
C_{23}	Customer Service	1Ku_96oz	0.82
C_{23}	Plant Maintenance	1In_b2z3	0.92
C_{23}	Plant Maintenance	1In_avon	0.91
C_{23}	Plant Maintenance	1In_ajlf	0.90
C_{23}	Plant Maintenance	1In_bd6i	0.87

(c) Topic 1 (order) in real-world processes from a large IT service company

cls id	category	process title	prob.
C_{11}	FIN	Process Purchase Requisitions	0.79
C_{11}	FIN	Manage Material Master	0.91
C_{11}	FIN	Process Purchase Orders	0.96
C_{11}	FIN	Process Good Receipts	0.80
C_{12}	O2C	Order-Donation Order	0.99
C_{12}	O2C	Order-Front End Order	0.94
C_{12}	O2C	Process Sales Order	0.98
C_{12}	O2C	Order-ESW Order or Bump	0.95
C_{12}	O2C	Order-Credit & Debit Memo Request	0.99
C_{12}	O2C	Order-Trial Order	0.99
C_{12}	O2C	Order-BP Stock Order	0.97
C_{12}	O2C	Order-Multiple Business Models	0.98
C_{12}	O2C	Order-CSC & ROC Order	0.99
C_{13}	MDM	Add a New Location	0.93
C_{13}	MDM	Register New Country Enterprise	0.79
C_{14}	O2O	Reconcile Inventory	0.62
C_{14}	O2O	Manage Sell Out Sell Thru Proc	0.69
C_{15}	O2C	Manage Returns and Disposition	0.95
C_{15}	O2C	Manage Outbound Delivery	0.95

In summary, we evaluated our two-level business process clustering method qualitatively. The experimental results indicate that our method can effectively discover some underlying information of process repositories and cluster processes with similar characteristics together.

4.2 Business Processes Retrieval Evaluation

In this section, we compare four business process retrieval methods introduced in Section 3.2, i.e., *LDA-retrieval*, *LDA-cluster-retrieval*, *kmeans-cluster-retrieval*, and *structure-matching-retrieval*[4], which is the baseline retrieval method.

Mean average precision (MAP) is used to measure the retrieval performance. Given a query process, the average precision is calculated as $\sum_{j=1}^{n}(precision[j] \times rel[j])/R$, where R is the number of relevant processes in the repository and $precision[j] = \sum_{k=1}^{j} rel[k]/j$. If the jth process is relevant to the query, $rel[j]$ is one, otherwise zero. The mean average precision is the mean of average precisions over a set of queries. To calculate MAP, we first need to identify all the relevant processes given a particular query. This is not a trivial work for a large process repository. A pooling approach used by TREC [18], is employed to get the complete relevance information, where only processes that are returned as possible answers to a query by the participating retrieval systems are evaluated while all other processes in the repository are assumed to be not relevant. Specifically, for each query, we combine the top 20 returned processes from all the retrieval systems and evaluate if these processes are relevant or not. All the other processes that are not retrieved are assumed to be not relevant. This manual evaluation was conducted independently by three authors of this paper adopting a majority voting[5].

Parameters. There are two parameters in *LDA-retrieval* that need to be determined, i.e., the weight γ for the LDA model and the Dirichlet smoothing parameter μ in (2). All the other calculations or parameters can be directly obtained from the LDA model in the business process clustering step. We use the SAP reference repository as training data to select the parameters γ and μ. The other data sets are used to test whether the selected parameters can be applied consistently. In *LDA-retrieval*, we perform a "grid search" on γ and μ. Various pairs of (γ, μ) are tried and the one yielding the best MAP is selected, i.e., $\gamma = 0.2$ and $\mu = 400$. As the LDA model in the clustering step is used in *LDA-retrieval* and *LDA-cluster-retrieval*, the topic number of LDA may also have an influence on their retrieval results. We tried these two LDA related retrieval methods on the SAP reference repository with different number of topics. When the topic (aka cluster) number is equal to 20, the same setting of the LDA model in the clustering step, these two methods are able to achieve close-to-best or good mean average precisions. Therefore, the parameters and results of the LDA model from the clustering step can be directly used in the retrieval to save computational cost. However, more careful parameters selection can be performed in order to obtain the best retrieval results. For comparison, the number of clusters in *kmeans-cluster-retrieval* is set to be the same as that in *LDA-cluster-retrieval*.

[4] We abuse structure matching slightly here since some data sets do not have structure information and the semantic similarity score is used.

[5] For SAP reference models, the evaluation was conducted by the first author of this paper due to the access constraints for the other authors.

Retrieval Results. We randomly select a set of query processes from each repository, specifically, 10 processes for SAP reference models and 5 for the other four data sets. Table 3 shows the mean average precisions of the top 20 retrieved processes for these systems over each query set. The highest MAP achieved for each repository is in bold font. As we can see, *LDA-retrieval* has the best retrieval performance over all the data sets except for SAP best practice (cross industry). Different from *structure-matching-retrieval*, in *LDA-cluster-retrieval*, only a subset of processes are retrieved and compared against to the query in terms of their structure similarities. For most repositories, *LDA-cluster-retrieval* has a higher MAP than that of *structure-matching-retrieval*. The reason is that some relevant processes may not share any partial structures with the query, i.e., they are relevant in terms of high level text similarities. These processes are very likely to be assigned into the same cluster with the query and therefore get retrieved. On the other hand, some processes sharing partial structures with the query may not be relevant. Similarly, *kmeans-cluster-retrieval* also outperforms *structure-matching-retrieval* for some data sets, but is inferior to *LDA-cluster-retrieval* in general. Note that the low MAP by *structure-matching-retrieval* for APQC is due to the fact that there are not many common steps among processes in this repository. The structure similarities between most processes in APQC and the query are zero and therefore *structure-matching-retrieval* can only randomly retrieve some processes to put in the top 20 list. Our *LDA-retrieval* method significantly outperforms *structure-matching-retrieval* for all the evaluated data sets.

Table 3. Mean average precisions of four retrieval systems on different process repositories

MAP	SAP (industry)	SAP (cross industry)	SAP reference	APQC	Real practice
Structure matching	0.1968	0.4396	0.5771	0.0222	0.3125
LDA-retrieval	**0.4615**	0.4725	**0.7940**	**0.4076**	**0.5550**
LDA-cluster-retrieval	0.2311	**0.5034**	0.5692	0.0986	0.3636
Kmeans-cluster-retrieval	0.1976	0.3934	0.5019	0.0337	0.3676

Running Time Analysis. Table 4(a) lists the total running time of four retrieval systems over all the queries for each repository. All these systems are evaluated on a laptop with a dual core 2.66 GHz Intel CPU, 4.00 GB RAM, and Java Virtual Machine version 1.6 (with 256MB of heap size). For SAP reference models, *structure-matching-retrieval* takes up to 833.31 seconds while the other methods use a significantly smaller time. For the other data sets, the running time difference among these methods is quite small due to simple structure or semantic similarity computations. Since these retrieval systems also depend on the first level clustering results, we also show the running time of LDA and kmeans clustering for different process repositories in Table 4(b). The model inference of LDA is much more complicated than kmeans, so LDA clustering takes a longer time. Note that for the SAP reference models, even though we add the running time of LDA clustering to the retrieval for these methods, they still cost a much smaller time than *structure-matching-retrieval*. In practice, process clustering only needs to be conducted once for each data set. After that, the process clusters and trained LDA model can be readily used for future retrieval.

Table 4. Running time of process retrieval and the first level clustering (seconds)

(a) Running time of four retrieval systems on different process repositories

Running time	SAP (industry)	SAP (cross industry)	SAP reference	APQC	Real practice
Structure matching	0.025	0.018	833.31	0.013	0.014
LDA-retrieval	0.12	0.15	69.02	0.042	0.048
LDA-cluster-retrieval	0.005	0.003	91.72	0.004	0.005
Kmeans-cluster-retrieval	0.008	0.011	108.93	0.016	0.010

(b) Running time of LDA and kmeans clustering on different process repositories

Running time	SAP (industry)	SAP (cross industry)	SAP reference	APQC	Real practice
LDA-clustering	36.59	28.66	54.67	5.73	10.51
Kmeans-clustering	2.03	2.71	1.23	0.31	0.23

5 Conclusions and Future Work

In this paper, we present a business process clustering and retrieval method that combines language modeling and structure matching approaches in a novel way. While clustering of business processes helps reveal the underlying characteristics and commonalities among a given set, language modeling based retrieval helps discover the top k most similar processes to a given query more efficiently than the pure structure matching approach. We leverage both the text terms in process step names and the process documentation. In the absence of any formal models, processes can still be matched based on textual/topical similarities. This ability to work with different kinds of documentation about business processes makes our approach more practical since the quality of documentation for real-world business processes can vary widely. Our current experimental evaluation is "homogeneous" in the sense that the clustered or retrieved processes are from the same repository. In the future, we can apply the proposed method to "heterogeneous" clustering and retrieval within a pooled process repository. Processes with different modeling notations can be efficiently matched based on their text features whereas the pure structure matching involves a lot of conversion efforts in the first place. The *compound* data type of business process models requires novel clustering and retrieval methods which can incorporate process information with different modalities, for instance, control flow, data flow, detailed design documentation, and execution logs. In addition, work from process mining area which derives process model from execution logs can be combined with other forms of available documentation to enhance process similarity matching. We believe that instead of relying on a single type of information about processes, considering multiple sources of information will yield better insights about business process similarities.

References

1. Akkiraju, R., Ivan, A.: Discovering Business Process Similarities: An Empirical Study with SAP Best Practice Business Processes. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 515–526. Springer, Heidelberg (2010)

2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
3. Croft, W.B.: A model of cluster searching based on classification. Information Systems 5, 189–195 (1980)
4. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph Matching Algorithms for Business Process Model Similarity Search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
5. Dong, X., Halevy, A.Y., Madhavan, J., Nemes, E., Zhang, J.: Simlarity Search for Web Services. In: Proc. of VLDB 2004, pp. 372–383 (2004)
6. van Dongen, B.F., Dijkman, R., Mendling, J.: Measuring similarity between business process models. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 450–464. Springer, Heidelberg (2008)
7. Dumas, M., García-Bañuelos, L., Dijkman, R.M.: Similarity Search of Business Process Models. Bulletin of the Technical Committee on Data Engineering 32(3), 23–28 (2009)
8. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: Proc. of APCCM 2007, pp. 71–80 (2007)
9. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proceedings of the National Academy of Sciences 101(suppl. 1), 5228–5235 (2004)
10. Grigori, D., Corrales, J.C., Bouzeghoub, M., Gater, A.: Ranking BPEL Processes for Service Discovery. IEEE T. Services Computing 3(3), 178–192 (2010)
11. Jung, J., Bae, J., Liu, L.: Hierarchical clustering of business process models. International Journal of Innovative Computing, Information and Control 5(12), 1349–4198 (2009)
12. Keller, G., Teufel, T.: SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping. Addison-Wesley, Reading (1998)
13. Li, J.: Two-scale image retrieval with significant meta-information feedback. In: Proc. of ACM Multimedia Conference 2005, pp. 499–502 (2005)
14. Liu, X., Croft, W.B.: Cluster-based retrieval using language models. In: Proc. of ACM SIGIR Conference 2004, pp. 186–193 (2004)
15. Madhusudan, T., Zhao, J.L., Marshall, B.: A case-based reasoning framework for workflow model management. Data Knowl. Eng. 50(1), 87–115 (2004)
16. Ponte, J., Croft, W.B.: A language modeling approach to information retrieval. In: Proc. of ACM SIGIR Conference 1998, pp. 275–281 (1998)
17. Salton, G., Wong, A., Yang, C.S.: A Vector Space Model for Automatic Indexing. Communications of the ACM 18(11), 613–620 (1975)
18. Turpin, A., Scholer, F.: User Performance versus Precision Measures for Simple Search Tasks. In: Proc. of ACM SIGIR Conference 2006, pp. 11–18 (2006)
19. Voorhees, E.M.: The cluster hypothesis revisited. In: Proc. of ACM SIGIR Conference 1985, pp. 188–196 (1985)
20. Wei, X., Croft, W.B.: LDA-based document models for ad-hoc retrieval. In: Proc. of ACM SIGIR Conference 2006, pp. 178–185 (2006)
21. Yan, Z., Dijkman, R., Grefen, P.: Fast Business Process Similarity Search with Feature-Based Similarity Estimation. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 60–77. Springer, Heidelberg (2010)
22. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to ad hoc information retrieval. In: Proc. of ACM SIGIR Conference 2001, pp. 334-342 (2001)