Information Resource Recommendation in Knowledge Processes

Tadej Štajner, Dunja Mladenić, and Marko Grobelnik

Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
{tadej.stajner,dunja.mladenic,marko.grobelnik}@ijs.si

Abstract. This paper proposes a framework and an implementation for pro-active just-in-time information resource delivery, based on knowledge processes. We focus on providing a desktop application that presents a ranked list of information resources, such as documents, web sites or e-mail messages, that are considered to be most relevant in that point in time. The paper decomposes the recommendation problam into subproblems and provides evaluation on several event, action and process models. Results show that defining actions based on clustering of events yields the best recommendations.

Keywords: Process modeling, Information delivery, Clustering, Process mining, Just-in-time Information Retrieval.

1 Introduction

Given the scenario of knowledge workers in an enterprise whose workflows consist of accessing various resources, such as web pages, documents and e-mail messages, we propose a prototype for process-based prediction and its implementation. The premise of this work is that we can learn a model from the knowledge worker's use of information resources. We refer to that model as the knowledge process model. We use the learned model with the purpose of streamlining the workflow by suggesting information resources to the user before he needs to retrieve them [10]. For the purpose of this paper, we use the term *knowledge process model,* which corresponds to a set of patterns that describe how a knowledge worker is using various information resources [1]. We construct it by learning from existing event logs under the assumption that the event logs are generated by an underlying knowledge process model.

We capture these events in an on-line fashion through workspace instrumentation and logging infrastructure. For example, the events consist of visiting search engines, web sites, documents on the desktop and receiving and sending e-mail. We use the resulting event stream to learn the model, as well as react with suggestions for information resources given the live data. The data in our model is constructed from three basic components: text, social network, and time.

We decompose the problem into subproblems that form a *knowledge process framework* that enables us to consider process mining on top of primitive events. We outline various *event*, *action* and *process* models that fit various scenarios and

R. García-Castro et al. (Eds.): ESWC 2011 Workshops, LNCS 7117, pp. 186–193, 2012.

patterns that we have encountered in our experiments. The *event models'* role is feature construction: fitting sequences of primitive events with various flavors of content, semantics and social network information and encoding them into feature vectors. The *action models* take these sequences of feature vectors and transform them into sequences of actions, which can them be processed by *process models* to act as predictors of most likely future actions, from which we can then apply a ranking on a candidate set of information resources. Information resources should have a representation compatible with the representation of the primitive events.

2 Related Work

Process mining is often used to uncover dependencies, control flows and patterns in a given business process execution log, encoding the model in eEPC diagrams or Petri nets [2]. Whereas these sort of approaches were built for the purpose of discovering process knowledge that enabled organization optimization, we are targeting the use case of pro-active information delivery. As presented in Holz et al. [3], such approaches mainly vary in two dimensions: process support and information delivery, meaning that some information delivery approaches work on top processes ranging from ad-hoc to strictly-structured on one dimension, and information delivery mechanisms ranging from light-weight to heavy-weight. In our domain, having light-weight modeling is important to simplify deployment and adaptation to various knowledge worker use cases. Our design goal is to maximize information delivery performance while learning from usage logs alone, without any user supervision. Besides enterprise knowledge processes, systems that learn while monitoring were also proven successful in the e-learning domain [4] with work-integrated learning [5].

3 Knowledge Process Model Framework

We have designed and implemented a framework for prediction on top of knowledge processes that solve issues that we have encountered with designing recommendation use cases on top of classic process mining models. Process mining expects pre-defined atomic actions as its input – each event unambiguously representing an action. This requirement may not be easily satisfied in some domains, such as knowledge work [6], where a lot of activity is ad-hoc and does not follow a prescribed process. The data that we are dealing with is a natural example of the *TNT* (text, network, time) framework [8]: the data points are events, carrying temporal information, content and a social network component. To fit into that model, we use a framework that is designed to handle this domain to support process mining on top of semi-structured data [7].

We decompose the information delivery problem in knowledge processes into a framework with three separate subproblems, each solveable with multiple approaches. To better illustrate the design decision behind the three distinct steps, let us start from the final step: the *process model*. In business process modeling, a process model describes patterns of atomic actions and is able to provide us with a probability estimate that one action will follow another. In other word, a process model maps from an action history to a probability distribution across possible following actions.

Given that our input is in form of primitive events, we need to transform the TNT events into actions – using an *action model*. However, directly constructing an action model to map from TNT events to process actions is not always possible nor practical, since the content within the events may not have the same properties across different domains. For this purpose we define the *event model*, which represents the feature construction phase from the raw TNT events.

The prediction scenario uses all of these three steps in the following setting: given a user's history of TNT events and possible resources that will be used, transform both the history and the future candidates into actions. Next, use the process model to evaluate each candidate's probability of appearing given the observed history. This information is then used in evaluation as a ranking score for information resources.

3.1 Feature Construction with Event Models

TNT events may contain the information on the actor, textual content, social network information, event metadata and the time of execution. We partition the log of events into sessions, which represent instances of knowledge process executions.

A **vector space event model** is represents the TNT event as a vector of features, derived from the event's text, social network and metadata. For text-derived features, we apply a TF-IDF weighing scheme.

This model can be extended by also encoding the information from neighbouring events. The sequential addition to the simple vector space event model enables event representations that captures some temporal dependencies, resulting in a **session vector space** model, which takes into account the fact that another has been executed in the same session within a given window.

3.2 Action Models

As basic process models still map from a space of action-to-action transitions to conditional probabilities, we still need to map sequences of documents into sequences of actions.

Independent Feature Actions. There are several ways to avoid high dimensionality of actions: one is to treat event features independent and consider them as individual actions themselves. However, this leads us to an issue where we have multiple possible actions per event, which we need to interpret. We solve this in the following fashion: given a pair of consecutive event vectors, consider at each possible feature pairing from these two vectors as a potential sequence. For instance, given a sequence of two consecutive events $d_1 = \{a = 0.7, b = 0.7\}$ and $d_2 = \{x = 0.7, y = 0.7\}$, we translate this to action sequences of two consecutive actions: *(a,x)*, *(b,x)*, *(a,y)* and *(b,y)*. In other words, each of these action sequences is a possible interpretation of (d_1,d_2) in this action model. Since the operation considers all possiblities, we only consider the immediate neighboring event. The consequence of this approach is that for some concrete dataset, the feature space of events is the same as the feature space of actions. This is one way to model the fact that an event may have many features. This model provides a tradeoff that keeps the space of possible actions reasonably low-dimensional, but assumes conditional independence of features.. Where multiple actions may represent a single event. To resolve this, we choose the most likely interpretation.

Clustering-Based Actions. As has been suggested in [6], one way to define actions is by performing various types of clustering on the event feature vectors while still retaining high predictive power [7]. This model allows us to control the dimensionality of the action space by varying the *k* parameter in clustering. Upon mapping events into actions, we classify the new set of events into actions via a centroid classifier.

3.3 Process Models

The core of the framework are the process models. They estimate the probability that a certain action will take place conditioned by the last couple of observed actions.

Smoothing of the process model is necessary because sparseness of data. In order to successfully compute a probability of a sequence, none of the sub-sequence probabilities must turn out to be zero. However, if there is no evidence for a particular action following a particular sequence, it does not mean that that sequence will also not occur in test data. For that purpose we employ Laplace add-one smoothing.

Even though the framework allows any possible combination of event, action and process models, some combinations yield better results than others. This framework enables two different ways to encode dependencies in the knowledge process: either via features in sequential- and session-based event models, or explicitly within the core process model that models conditional probabilities between successive actions. We allow and experiment on both scenarios, since interesting dependencies might exist either on the level of individual features, or between actions, having a higher level of abstraction.

4 Implementation

We have implemented prediction that fits into the desktop environment of knowledge workers. The scenario is the following: imagine a desktop widget that tracks the current state of the user's workspace with most recently accessed resources. The widget itself is rendered as a ranked list of information objects that are considered most relevant for the user. We use the following semantic properties as features for individual events: bag-of-words of document content, social roles of participants (inside vs. outside of organization, manager, developer, researcher, private vs. multiple people, single vs. multiple organizations) and event metadata (type of event, type of media).

For cases where the user has just started his session and does not have any recently accessed resources or there is are no possible predictions from the process model, we use multiple criteria. We ranking using the following criteria:

- The probability of the candidate information resource d_i given the history: $P(d_i|d_{hist\ 0}..., d_{hist\ j})$;
- The similarity of the candidate information resource to the most recently observed information resource: $sim(d_i,d_{hist\ j})$;

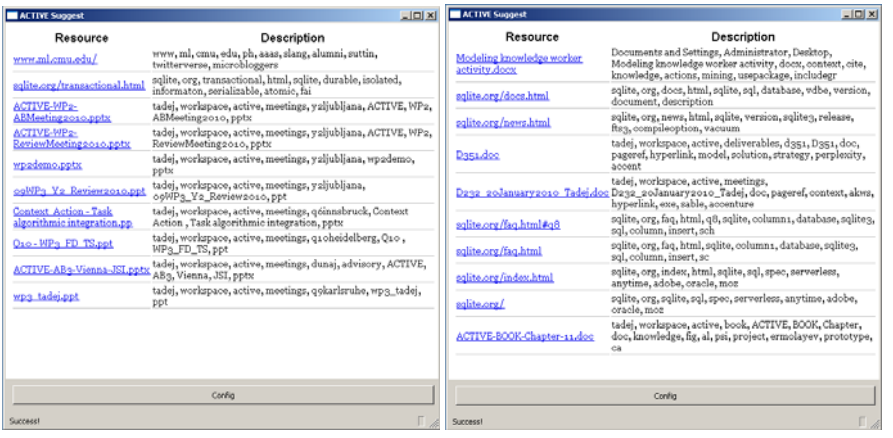

Fig. 1. Examples of recommendations for various situations in the knowledge process

The implemented front-end is a graphical user interface, displaying an ordered list of most probable information objects given the current state of the user, as shown in Figure 1. Given the live events coming from the monitoring infrastructure, the information delivery framework converts these events into feature vector documents and then action sequences which represent the user's session history. Then, it selects a set of possible interesting information resources with standard text retrieval techniques and evaluates the probability of each known information object given the observed history using the knowledge process model recommender.

Figure 1 shows some examples of recommendations, offered by the service. The first image shows the state of the recommender after reading a documentation of a back-end system for process mining, which is reflected by similar content types being offered. The second image demonstrates another point in the knowledge process, showing recommendations after opening a technical presentation on machine learning. It offers various related documents, as well as web sites that might contain useful information in that situation which have been accessed in similar situations.

5 Experiments

In our experiments, we have performed evaluation by constructing combinations of models with one set of data and evaluating them with another subset using ten-fold cross-validation to account for generalization error. We ran the experiment ten times, using nine partitions as training data and the remaining one as a test dataset. Given this setup, we can observe several evaluation metrics, relevant to evaluation of models. The dataset of 31882 events was gathered by monitoring three knowledge workers in a telecommunications company.

We evaluated various knowledge process model configurations by measuring how high do they rank the actual events in the test set. In order to conduct this experiment, we simulate the prediction behaviour by querying the system with partial sequences to which we already know the correct TNT event that will follow and then check how

high does that particular event rank in the prediction output. In other words, we are measuring how well can this system predict the actual workflow. We then report the average reciprocal rank for a particular combination of models and average number of predictions that score in the top 20, which are the key indicators in our experiment.

We have evaluated on the knowledge worker logs. Since the TNT events have no pre-defined actions, we use the vector space event models – *IDF* and *SessionIDF*. They all use the TF-IDF weighing scheme for features, but differ in the way that the features are generated: whereas *IDF* contains only the features from a particular event, *SessionIDF* also encodes between the features of the events in the same session. For process models, we experimented on either using an identity transform which assumes uniform processes and acts as a baseline without a process model or a bi-gram Markov model with Laplace smoothing, modeling the dependencies between successive events.

Table 1. Experiment results with various combinations of event, action and process models for the desktop use case

Event Model	Action Model	Process Model	Reciprocal rank	Percentage in top 20
IDF	Independent	None	0.0612	0.2220
IDF	Independent	Laplace	0.0803	0.2377
IDF	Clustered:10	None	0.0794	0.2697
IDF	Clustered:10	Laplace	**0.1076**	**0.3485**
IDF	Clustered:30	None	0.0853	0.3081
IDF	Clustered:30	Laplace	0.0797	0.2490
SessionIDF	Independent	None	0.0774	0.2895
SessionIDF	Independent	Laplace	0.0750	0.2674
SessionIDF	Clustered:10	None	0.0756	0.2807
SessionIDF	Clustered:10	Laplace	0.0701	0.2384
SessionIDF	Clustered:30	None	0.0832	0.3013
SessionIDF	Clustered:30	Laplace	0.0874	0.3051

We have ran the experiments on the dataset, gathered in two separate two-month time periods, totaling 31882 events. Here, the events consist of web browsing events, editing of Microsoft Office documents and working with e-mail. Results, shown in Table 1, show the following: best performing configuration is the one with plain TF-IDF features of events, defining a clustered action model with few distinct clusters as actions and a process model with Laplace smoothing. All in all, we are able to place the correct information resource in the top-20 list roughly on over one third of occasions.

6 Conclusion

An important lesson is that many of the issues in process mining lie in transforming the input data into well-defined actions. By developing a framework that integrates the data transformation steps within the mining process itself, we can simultaneously solve the issue of transforming events into atomic actions, as well as learning process models, reducing the preprocessing requirements for implementing a predictive

application. We have discovered that encoding the knowledge process patterns in the feature vectors using the session-based feature construction is a valid approach, showing significant improvement from the baseline while having a very simple implementation. However, our use case shows that using an explicit process model can ultimately outperform session-based features in events. Results show that assuming independence of features does not provide desirable performance.

Implementation-wise, we have produced a simple user interface that does not require that the user interacts with it in any special way to train it, lowering the barrier to practical usage. For our future work, the clustering-based action definition will include using a complex graph representation of data so that we can avoid flattening the semantic network structure into event features. To take advantage of the structural information and to correctly handle differences in distributions across people, events and resources, we will represent the data in a relational representation and employ multi-relational clustering algorithms which are able to handle such representations. We will also experiment in applying this model in other specific knowledge worker domains, focusing on supporting multiple knowledge workers with personalized models. Future work on refining recommendation will also focus on constructing more personalized models by constructing a general model combined with a personalization layer. In terms of incorporating external knowledge, we will consider semi-supervised learning from user feedback, as well as employing pre-existing process models in combination with learned process models. Further evaluation will also consider sensitivity of a recommendation. In real scenarios, the benefit of a recommendation may be negated by the interruption of the user's workflow.

Acknowledgments. This work was supported by the Slovenian Research Agency and the IST Programme of the EC under ACTIVE (IST-2008-215040), ALERT (ICT-249119-STREP) and PASCAL2 (IST-NoE-216886).

References

1. Warren, P., Kings, N., Thurlow, I., Davies, J., Bürger, T., Simperl, E., Ruiz, C., Gomez-Perez, J., Ermolayev, V., Ghani, R., Tilly, M., Bösser, T., Imtiaz, A.: Improving knowledge worker productivity the ACTIVE approach. BT Technology Journal 26(2) (2009)
2. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W(E.), Weijters, A.J.M.M.T., van der Aalst, W.M.P.: The ProM Framework: A New Era in Process Mining Tool Support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)
3. Holz, H., Maus, H., Bernardi, A., Rostanin, O.: From lightweight, proactive information delivery to business process-oriented knowledge management. Journal of Universal Knowledege Management 2, 101–127 (2005)
4. Lokaiczyk, R., Faatz, A., Beckhaus, A., Goertz, M.: Enhancing just-in-time e- Learning through Machine Learning on Desktop Context Sensors. In: Kokinov, B., Richardson, D.C., Roth-Berghofer, T.R., Vieu, L. (eds.) CONTEXT 2007. LNCS (LNAI), vol. 4635, pp. 330–341. Springer, Heidelberg (2007)

5. Rath, A.S., Devaurs, D., Lindstaedt, S.N.: Studying the Factors Influencing Automatic user Task Detection on the Computer Desktop. In: Wolpers, M., Kirschner, P.A., Scheffel, M., Lindstaedt, S., Dimitrova, V. (eds.) EC-TEL 2010. LNCS, vol. 6383, pp. 292–307. Springer, Heidelberg (2010)
6. Štajner, T., Mladenić, D.: Modeling Knowledge Worker Activity: Workshop on Applications of Pattern Analysis, Cumberland Lodge (2010)
7. Štajner, T., Mladenić, D., Grobelnik, M.: Exploring Contexts and Actions in Knowledge Processes. In: Proceedings of the 2nd Workshop on Context, Information and Ontologies (2010)
8. Grobelnik, M., Mladenić, D., Ferlež, J.: Probabilistic Temporal Process Model for Knowledge Processes: Handling a Stream of Linked Text. In: Proceedings of SiKDD 2009 Conference on Data Mining and Data Warehouses (2009)
9. Gomez-Perez, J., Grobelnik, M., Ruiz, C., Tilly, M., Warren, P.: Using task context to achieve effective information delivery. In: Proceedings of the 1st Workshop on Context, Information and Ontologies, pp. 1–6. ACM (2009)