Towards a Knowledge-Intensive and Interactive Knowledge Discovery Cycle

Benoît Mathern\textsuperscript{1,2}, Alain Mille\textsuperscript{1,3}, Amélie Cordier\textsuperscript{1,3}, Damien Cram\textsuperscript{1,3,4}, and Raafat Zarka\textsuperscript{1,5}

\textsuperscript{1} Université de Lyon, F-69622, Lyon, France
\textsuperscript{2} IFSTTAR, LESCOT, F-69675 Bron, France
\textsuperscript{3} Université Lyon 1, CNRS, LIRIS, UMR5205, F-69622, France
\textsuperscript{4} myTwatch, F-75002, France
\textsuperscript{5} INSA-Lyon, CNRS, LIRIS, UMR5205, F-69622, France

Abstract. Knowledge capture and reuse is a challenging task consisting of many steps. The knowledge discovery cycle presented by Fayyad \cite{Fayyad:1996} offers a global overview of how these steps are combined together. By taking a step back and considering data as activity traces, we propose to change this knowledge discovery cycle. We consider that knowledge is built rather than discovered. Therefore human involvement to make sense out of the traces is paramount. We propose to use knowledge engineering techniques to benefit from the user’s knowledge as early as possible. The use of modelled traces can solve this issue and change the knowledge discovery process to make it more interactive and knowledge-intensive. The proposed methodology will be illustrated with software applications we have developed which, combined together, support the whole process of knowledge discovery. A discussion of the proposed methodology and of the required tools is given.

Keywords: Interactive Knowledge Discovery, Trace-Based Systems, Human-Computer Interaction

1 General Introduction

Learning from experience requires to capture experience in such a way it can be used for further action. It is difficult to acquire such knowledge (e.g., rules or cases) from experts. In this paper, we propose to combine knowledge engineering and knowledge discovery (KD) approaches to acquire such knowledge. KD aims at discovering knowledge from data. We propose to adapt KD in the context of activity traces. Our goal is to allow to capture and apply knowledge coming from both traces and human expertise. Therefore, the general goal of this work is what could be called “man machine interactive knowledge discovery”.

The first issue is to get relevant data among what is available in the environment and to prepare these sources to build a simple model of observed elements. Another issue is to offer interactive tools to support a dynamic knowledge discovery process: thus discovering new knowledge from the sequences of observed
elements. A third issue is to take into account what has already been discovered, and what is already known to guide the KD process. Lastly, it is important to be able to provide actionable knowledge for humans and machines.

Discovering knowledge by observing processes, their behaviour and their productions is not new. There is a strong tradition in the Human Computer Interactions research \cite{17} to propose concrete methods for discovering explicit knowledge from various observation sources (computer events, video and audio records, data, texts, etc.). We can use the general notion of “interaction traces” to refer to such knowledge sources and we are specifically interested in exploiting traces of events for supporting the Knowledge Discovery of the dynamics of processes. In order to support our claim several points will be discussed in this paper. First we provide a brief overview of the classical Knowledge Discovery Cycle and of its challenges. We consider KD to be a Knowledge Intensive task, and we focus on the specific issue of discovering knowledge from the observation of behaviours. Then, we develop our claim that Trace-Based Man Machine Interactive Knowledge Discovery should be a good approach for dynamic knowledge engineering and would make the reuse of knowledge easier by providing in-depth interaction in the KD process. Examples of softwares are introduced to illustrate a possible general framework which implements the proposed dynamic knowledge engineering approach. The relevance of this approach is then discussed in regards to the feedback obtained from practical knowledge discovery experience. The paper concludes with a discussion about future work and applications.

2 The Classical Knowledge Discovery Cycle

In the early 90’s the use of data mining tools in order to discover new knowledge from data led to the development of knowledge discovery in databases (KDD). Frawley et al. defined knowledge discovery as “the nontrivial extraction of implicit, previously unknown, and potentially useful information from data” \cite{9}. This first definition stresses the fact that something new and useful must be discovered.

In 1996, Fayyad et al. proposed a formal description of the knowledge discovery process \cite{11}. The authors presented the KDD process as “interactive and iterative, involving numerous steps with many decisions made by the user” (Fig. \ref{fig:kdd_cycle} shows an overview of this cycle). With this cycle the authors showed that data mining is only one step of the knowledge discovery process. In order to discover knowledge, all the steps of the process are important. More importantly, the knowledge discovery process is guided by a user – generally an analyst. The role of the user is paramount as the user sets the goal of the discovery, defines the parameters at each step, and interprets the discovered patterns.

2.1 Knowledge Discovery Challenges

Data Mining and Knowledge Discovery (KDD) Challenges. The KDD community listed a number of challenges \cite{8} and the most important of them
Before any discovery, the user gathers domain knowledge and identifies the goal of the knowledge discovery task. Then, the user creates a data sample (target data) from which to discover knowledge. Then, the user cleans, preprocesses and transforms data to prepare the mining. Then, the user selects a data mining method, and sets its parameters. Afterwards, the user interprets the discovered patterns. Finally, the discovered knowledge can be documented, or used to fulfill the goal initially set by the user. Note that at any step the user can go back to any previous step of the cycle.

The original process centric view of KDD espoused the three “I”s (Integrated, Iterative, and Interactive) as basic for KDD. These are central to the ideas of “Computer Assisted Human Discovery” and “Human Assisted Computer Discovery.” There has been very little work on these in recent years.

**Actionability.** Researchers have tried to address these challenging issues by using complementary approaches to produce *actionable knowledge*. The notion of actionable knowledge has been defined to estimate the usefulness of discovered knowledge, with a measure of its actionability. A knowledge is actionable if it can actually be used by the end user (i.e., to make decisions). Cao and Zhang proposed a practical methodology to discover actionable knowledge by integrating domain knowledge to guide the data mining process. Yang et al. addressed the issue of providing actionable knowledge by learning action models (expressed in PPDL) in order to make the discovered knowledge available for a planner and to express the learned structure in such a way that it is understandable for an expert in planning. Here, actionability actually provides a planner or an expert with what is needed to build new plans.

---

6 **PPDL:** Planning Domain Definition Language.
Knowledge Discovery as a Knowledge Intensive Task. Brachman et al. [2] offer a practical overview of what knowledge discovery actually implies for the user. The authors focus on the importance of the involvement of the user in the discovery process. They propose to redefine knowledge discovery as “a knowledge-intensive task consisting of complex interactions, protracted over time, between a human and a (large) database, possibly supported by a heterogeneous suite of tools”. This definition highlights the main difficulties of knowledge discovery: knowledge discovery is knowledge-intensive, requires interactions, and the tools often lack integration. Our position paper consider this challenge as the central one for guiding our proposals, and explaining our first results.

2.2 Discovering Knowledge from the Observation of Behaviours

Knowledge Discovery in Databases implies the idea that data intrinsically contains information available for an analyst to interpret. In the KDD process, data is an entry point. However, data is a result of the process of data collection. What users actually care about is not the data itself, but what the data represents in the real world. For this reason, we prefer to use the term of Knowledge Discovery (KD) as the capture of knowledge has an important role to play in the discovery process.

Data is obtained from a two step process. First, a behaviour or process occurs in the real world. To avoid confusion, we refer to such a process as an activity. Second, this activity is observed and data is captured, leading to the creation of databases. The goal of knowledge discovery is to discover knowledge about the observed activity.

In this paper the application domain used to illustrate the KD process is the field of transport, with car driving as the activity of interest. The goal of KD is to understand drivers’ behaviour and cognition. Discovered knowledge can be used to develop driving assistance systems and improve road safety.

The record of an activity is a trace of that activity. A trace typically contains information about the activity itself and its context. Traces also typically contain information about several activities, some relevant to the user and some that are considered by the user to be noise.

Often traces are fragmented, in the sense that the activity is recorded from different sources. Thus, the activity is not seen as a holistic process, but as several (fragmented) views of the same process. The goal of KD is to discover interesting knowledge about the activity. Therefore, the user needs to understand and transform the traces in order to, in the end, interpret the information available in the data. It is only when patterns or pieces of information are interpreted by the user that knowledge is created. The general process of knowledge discovery can been seen as follows:

\[
\text{Activity} \xrightarrow{activity \ produces} \ Traces \xrightarrow{user \ interprets} \ Knowledge \xrightarrow{user \ transforms}
\]

Traces are not usually standard digital objects, although they have specific properties including temporal properties and a semantic interpretation of ob-
served elements (events). Considering traces as data for a knowledge discovery process is a non-trivial task that does not usually appear in the KD phases. The general task of analysing an activity through observed behaviours has been theorised by Sanderson et al. [17], who proposed the notion of the “transformation” of traces to move from a first simple interpretation (almost raw data coming from sensors) to the actionable knowledge level of abstraction. As analysts better understand what is observed, they can check their interpretation by transforming a source trace into a target trace where candidate knowledge (according to some hypothesis) is used to describe and explain the observed activity. In turn this process is repeated on the target trace, which can be used as a source trace for further interpretation. The analyst is at the heart of the discovery process and has to be assisted in his/her task. The analysis process is a knowledge intensive task and, when using data mining techniques, needs iterative steps, interactivity and integration of the different phases of the data mining task within the discovery process.

3 Trace-Based Man Machine Interactive Knowledge Discovery

We claim that to improve the efficiency of KD from behavioural observations, it is necessary to provide a knowledge intensive framework. Such a framework should not only allow for the integration of what is already known in the domain but should also allow the reintroduction of discovered knowledge in the discovery process for future iterations. Throughout the discovery steps, the tools supporting discovery should be interactive, allowing the user to select and parameterise mining algorithms, and to build (actionable) knowledge structures, with the support of heuristics and inference mechanisms.

Interactivity is the key issue since the discovering activity is really shared between the knowledge discovery framework and the user (expert-analyst, end user, etc.). The three “I”s of the literature (Integrated, Iterative and Interactive) can be summarised by different levels of interactivity. “Integration” means that the whole framework provides a unique way to interact at the knowledge level, e.g., tools are able to share data and knowledge representations. “Iterations” means that the user can interact with the system to connect the different phases of the discovery process, either to use the knowledge acquired to loop back to a previous phase, or to use it as an input for a next KD phase. “Interactivity” means that the user and the framework are “coupled” to achieve a common task corresponding to the particular KD phase, for example, the user shares knowledge with the algorithms (through appropriate knowledge representations, such as models or constraints).

In order to simplify such a process, we propose a general framework unifying knowledge representation with dynamic modelling. In this framework, any discovered knowledge can be used as a new model of the mined sources. Also, this framework keeps track of the transformations applied to the traces, which allows the reuse or adaptation of transformations on new traces, and offers a way
to return to the original trace. Moreover, considering the specific case of understanding an activity, we propose to consider automata as a way to represent a discovered behaviour. Such a knowledge representation supports actionability as it allows to use the discovered automata as signatures of behaviours. Automata can be actionable either by detecting matching behaviours in traces or by producing the modeled behaviour, when used in a simulation engine. In the following section we briefly present a way to implement this proposal by using Modelled Trace and Trace-Based Systems. This provides a unified way to manage domain knowledge and discovered knowledge throughout the phases of the KD framework.

3.1 Modelled Trace and Trace-Based System

In order to manipulate traces as formal objects, our research team developed the notions of modelled traces and of trace-based systems [18]. A modelled trace (M-Trace) is the combination of a model and of a temporal sequence of observed elements (called obsels). Obsels are events, each characterised by a date, a type, attributes and relations. The model describes the semantics of the types, attributes and relations, while the temporal sequence of obsels is the record of what has been observed. For instance, for the activity of driving, obsels are time-stamped events (the driver brakes, accelerates, then turns the steering wheel, etc.) and the model is an ontology of the observed behaviours. A Trace-Based System (TBS) is a framework for managing M-Traces that provides trace-oriented services including: collecting M-Traces, computing sequences satisfying a pattern in a specific M-Trace, transforming a M-Trace into another one for abstraction purposes (filtering, merging, reformulating obsels), and navigating through the transformation graph of a M-Trace.

3.2 Towards an Interactive Knowledge Discovery Cycle

Fayyad et al. [7] proposed a general process for discovering knowledge from data (see Fig. 1). We propose a slightly different approach where data is collected from the observation of an activity, and a TBS provides support in the preparation and the transformation phases of the KD process. Furthermore, following the general idea defended by Michalski [15], we propose to complete this knowledge discovery process by a computer-supported synthesis step by providing a formal knowledge representation able to be used by a reasoning mechanism. This proposition is outlined in figure 2. These knowledge oriented approaches offer feedback to the analysts at each step with formal knowledge representations. Thus, facilitating the interactions with the analysts so that they understand the results and control the full Knowledge Discovery process. This approach has been introduced and detailed in [5], [18] and [11].
Fig. 2. The top part of the figure describes a new point of view on the (interactive) knowledge discovery cycle. The use of $M$-Traces allows to explicitly describe the semantics of data and therefore supports the knowledge-intensive process of knowledge discovery. The bottom part of the figure shows examples of tools supporting the interactive knowledge discovery cycle and highlights their role in the process.

4 Examples of Software Systems

In this section software tools that are developed in the context of interactive knowledge discovery are presented. A brief overview of the BIND, ABSTRACT, AUTOMATA, SCHEME EMERGER and TSTORE softwares is provided. Figure 2 shows how these tools can be integrated to produce a full interactive knowledge discovery cycle.

4.1 Bind – Abstract

Data is first acquired by tools capturing what happens in the environment. This is done by instrumenting the data capture with the means of tracing systems and sensors. For example, an instrumented car (or a driving simulator) is used to collect the driver’s actions, the vehicle state and the environment [12, p. 70].

Once collected, the data can be processed using the BIND framework [12, p. 69]. BIND offers high level functionalities to analyse car-driving data, such as signal processing functions or the synchronised visualisation of collected data and videos. BIND also allows the discovery of markers in the numerical data. For instance, it can create event markers when a certain value in the data is reached. These markers can be seen as traces of the activity.
These traces are then imported into the ABSTRACT software [11]. ABSTRACT is a TBS, which means traces are represented as $\mathcal{M}$-Traces. A TBS can be used to prepare, transform and visualise sequences of events as $\mathcal{M}$-Traces at the relevant level of abstraction with explicit semantics (Fig. 3). The ABSTRACT framework has been developed and used to support the Knowledge Discovery process in the context of driver behaviour analysis [10]. In this context, a huge quantity of data is collected during driving sequences, from numerous sensors, video recordings, and observer annotations. Then, the analyst can use the ABSTRACT interface to progressively transform traces in order to discover relevant patterns according to some hypothesis about the driver’s behaviour.

Fig. 3. Graphical interface of the ABSTRACT software. The horizontal axis represents the time and the vertical axis is related to the level of reformulation of the traces. Each symbol represents an obsel (the shape and the colour depends on the type of obsel). The user can click on each obsel to display specific information about this event, such as its type and attributes. The user can display different views of the same trace.

4.2 AUTOMATA as a Support for Interpretation of the Dynamics of Processes

In ABSTRACT new concepts can be defined corresponding to pattern signatures identified in the $\mathcal{M}$-Traces. This is a first step in the general issue of considering “knowledge mining” instead of “data mining” according to Michalski [15], since this process enriches the formal representation of the activity. In order to go a step further, and to actually provide actionability, the AUTOMATA software builds a Petri-net of a process being observed in the traces. The produced Petri-net represents knowledge about the dynamics of the process and is actionable, as it can be used to understand the process or to simulate it. AUTOMATA therefore supports the interpretation task, with the help of a trace mining process focusing on the available occurrences of patterns satisfying a particular request.

In order to implement a knowledge-intensive process, a mining algorithm has been modified to make it interactive [14]. The user can therefore use his/her

7 For more details about ABSTRACT, see http://liris.cnrs.fr/abstract/
knowledge about the activity to support the mining process. The non-modified algorithm [1] assumes that the traces are complete, which is rarely the case on real data sets. The interactive version of this algorithm allows the analyst to incorporate his own knowledge to compensate for the lack of information in the traces.

4.3 Scheme Emerger

The Scheme Emerger platform aims at discovering chronicles occurring in activity traces [4]. A chronicle is a pattern structure describing time constraints between types of events. Scheme Emerger implements an interactive mining algorithm that allows the user to constrain the mining process (e.g., patterns that are required or need to be excluded). The discovered chronicles are displayed to the user in real time, allowing the user to quickly obtain first results and to adapt the mining parameters (Fig. 4). Scheme Emerger can be used to execute queries to check the presence of chronicles defined by the user.

Fig. 4. Graphical interface of the Scheme Emerger platform [4]. View 4 shows the resources of the project that are handled by the analyst: constraint database, chronicles, traces and queries. Tabs 1 and 2 are chronicle and query editors. View 7 displays the trace currently being analysed. The types of events present in the current trace are displayed in view 3. View 6 lists in real time the chronicles discovered by the interactive algorithm. View 5 displays the chronicle selected in view 6, while view 8 lists the occurrences of this chronicle in the current trace. Clicking on one of these occurrences changes the focus of the trace (view 7).
4.4 TStore

TStore is a Trace-Based Management System (TBMS). The role of a TBMS is to store, manage and transform \(\mathcal{M}\)-Traces. A TBMS realises the \(\mathcal{M}\)-Traces storage, querying and handling layer of a TBS. The originality of TStore resides in the transformation facilities on \(\mathcal{M}\)-Traces that it offers. In addition to simple transformations such as \(\mathcal{M}\)-Trace filtering, \(\mathcal{M}\)-Traces aggregation or segmentation, TStore can use Finite-State Transducer (FST) to transform traces \[21\]. Transducers are automata with specific “translation” capabilities. TStore uses FST to transform \(\mathcal{M}\)-Traces by replacing obsels matching the FST with more abstract obsels.

5 Discussion

Table 1 offers a synthesis of how the different tools discussed in this paper can be used to address the challenging problems of knowledge discovery. BIND, ABSTRACT and AUTOMATA offer a complete chain of knowledge discovery which has been used by a research team composed of engineers and psychologists in analysing and modelling car-driving behaviours from data collected on instrumented cars \[13\]. This experience provides feedback on the relevance of the interactive and knowledge-intensive discovery cycle and on the tools themselves.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BIND (v.1)</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ABSTRACT (v.0.2)</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>AUTOMATA (v.0.11)</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SCHEME EMERGER</td>
<td>+</td>
<td>-</td>
<td>++</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>TSTORE</td>
<td>++</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>++</td>
</tr>
</tbody>
</table>


The importance of integration seems fairly obvious. For example, when a specific pattern is discovered, the user wants to be able to visualise the occurrences of that pattern (typically when it is unusual). This is only possible with an integrated suite of tools.

By working with driving experts who use the KD cycle, we discovered that the process of defining the semantics (in ABSTRACT the ontology which constitutes the model of a \(\mathcal{M}\)-Trace) was an important part of the KD process – from a practical point of view. This is probably due to the fact that experts already have knowledge about driving and part of the process is to make this knowledge explicit, rather than just discovering knowledge from the data. The formalism of \(\mathcal{M}\)-Traces allows the system to actually be Knowledge Intensive, by combining the data with a model describing their semantics.
Moreover, having explicit semantics describing the data has allowed these experts to more precisely document the data. Having proper documentation of the data is important as it allows users to properly interpret the meaning of data. It is even more important when the KD process involves multidisciplinary teams, i.e., from the electrician who understands the sensors instrumenting the car to the psychologist who understands the behaviour of a driver.

Trace-Based Systems record the transformations that lead from one $\mathcal{M}$-Trace to another. This aspect offers a documentation of the KD process itself which allows the user to easily find how a result has been derived. The documentation of the KD process could be further improved if the activity of the user (of the KD tools) was recorded. The user would then be able to introspect his/her own practice and could also use KD tools to gain a better understanding of how to improve his/her work.

The knowledge-intensiveness is practically tackled with the $\mathcal{M}$-Traces. This knowledge is actionnable as it is explicitly described in models that can be used by algorithms. Using automata to represent knowledge offers even more actionability. Indeed, automata can be used as a model for simulation and for the user to make decisions, or, as has been seen with TSTORE software, automata can be reintroduced in the KD process to transform $\mathcal{M}$-Traces. Therefore, combining Automata (which allows the discovery of automata from traces) and TSTORE would improve actionability by providing new types of interaction and globally benefit the KD process.

Overall, the feedback received from users was positive regarding the proposed solutions to support interaction, integration, iterations and actionability. However, a precise methodology needs to be defined in order to precisely measure the relative importance of these concepts in a real KD process. Evaluating the relevance of KD tools itself is a challenging task, as it consists in evaluating generative artifacts [6].

6 Conclusion

In this paper, several challenges of knowledge discovery that remain unsolved have been discussed. In the context of analysing sequential data, we proposed to adapt the classical KD methodology in order to make it knowledge-intensive. Tools supporting interaction, integration, iterations and actionability have been presented. Further work on each of these dimensions is needed to improve the KD process.

Acknowledgements The authors are grateful to the reviewers who gave thorough feedback and raised interesting points of discussion.

References

5. Deransart, P.: Conception de traces et applications (vers une méta-théorie des traces) (2010), document de travail. [http://hal.inria.fr/inria-00443648](http://hal.inria.fr/inria-00443648)