Towards Software Experts for Knowledge Discovery
A Process Model for Knowledge Consolidation

Lauri Tuovinen
Biomimetics and Intelligent Systems Group, University of Oulu,
P.O. Box 4500, FI-90014, Oulu, Finland

Keywords: Knowledge Discovery in Data, Process Model, Metadata, Knowledge Representation, Artificial Intelligence.

Abstract: The process of knowledge discovery in databases (KDD) is traditionally driven by human experts with in-depth knowledge of the technology used and the domain in which it is being applied. The role of technology in this process is to passively execute the computations specified by the experts, and the role of non-expert humans is limited to a few narrowly defined special cases. However, there are many scenarios where ordinary individuals would benefit from applying KDD to their own personal data, if only they had the means to do so. Meanwhile, KDD experts are looking for more advanced tools and methods capable of coping with the challenges posed by big data. Both these needs would be addressed by autonomous software experts capable of taking on some of the responsibilities of human KDD experts, but there are several obstacles that need to be cleared before the implementation of such experts is feasible. One of these is that while there is a widely accepted process model for knowledge discovery, there is not one for knowledge consolidation: the process of integrating a KDD result with established domain knowledge. This paper explores the requirements of the knowledge consolidation process and outlines a process model based on how the concept of knowledge is understood in KDD. Furthermore, it evaluates the state of the art and attempts to estimate how far away we are from achieving the necessary technology level to implement at least one major step of the process in software. Finally, the options available for making significant advances in the near future are discussed.

1 INTRODUCTION

In 1996, knowledge discovery in databases (KDD) was defined as "the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" (Fayyad et al., 1996). The seminal KDD process model, presented in the same paper, consists of five major steps: data selection, preprocessing, transformation, data mining, and interpretation/evaluation. In the past 20 years, research in the field of KDD has advanced greatly, but arguably both the definition and the process model are still essentially correct and useful.

A very interesting and challenging area of KDD research now is automating the planning of the KDD process and the interpretation of the results, which are tasks that currently require human experts. The potential impact of new results in this area is high, but to achieve them, we need to gain a better understanding of this aspect of the process. Until now, the inability to delegate these tasks to a computer may not have been a very significant problem, but in recent years there has been an explosion in the quantity and diversity of digital data available for KDD. As a result of this, our ability to actualize the potential value of data depends on the ability of computers to assist us in a more intelligent and comprehensive manner. Intelligent KDD tools are also a necessity for individuals who want to extract useful knowledge from their personal data but lack the required technology expertise.

The focus of this paper is on the interpretation/evaluation step of the KDD process, which is such a vast and complex one that it would be more accurate to represent it as a process in its own right. It is as the outcome of this process that the result of the data mining step is established as "valid, novel, potentially useful, and ultimately understandable", making it justifiable to call it knowledge. In a word, the objective of the process is to *consolidate* the newly discovered knowledge, so we shall refer to it as the *knowledge consolidation* process.

The purpose of the paper is to take some initial steps toward the eventual automation of the knowledge consolidation process by exploring its theoretical foundations and requirements. More specifically, the paper will: **1.** analyze the definition of knowledge

Tuovinen L.
Towards Software Experts for Knowledge Discovery - A Process Model for Knowledge Consolidation.
DOI: 10.5220/0006500502240232
In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KDIR 2017)*, pages 224-232
ISBN: 978-989-758-271-4

in KDD to clarify the meaning of the various *conditions* to be satisfied; **2.** outline the *steps* that need to be taken in order to satisfy the conditions once the data mining task of the KDD process has been completed; and **3.** define a set of new *capacities* of KDD software that need to be developed in order to automate the execution of the steps.

A core concept that the paper examines is that of a *software KDD expert*, an intelligent KDD software tool or agent capable of planning and executing KDD tasks without the supervision of a human expert. This capability enables a software expert to assist humans in setting up and running KDD workflows, substantially increasing the potential value of KDD technology in the hands of both experts and non-experts. By studying the knowledge consolidation process and its requirements, the paper aims to develop an understanding of how software KDD experts can be transformed from a theoretical concept to reality.

The remainder of the paper is organized as follows: First, a review of related work is given in Section 2. This is then followed by an examination of the definition of knowledge in Section 3. Section 4 outlines a process model for the task of interpreting and evaluating a data mining result such that it satisfies the conditions prescribed by the definition. Section 5 describes the major capacities that a software KDD expert needs to have in order to carry out knowledge consolidation, and also identifies the principal technical requirements of the capacities and compares them with the state of the art. Section 6 discusses the findings, and Section 7 concludes the paper.

2 BACKGROUND

There are two major ongoing trends that make rethinking the KDD process necessary. One of these is what has been referred to as the data deluge: the seemingly boundless increase of the quantity of digital data available for collection and analysis. It has been estimated that a total of 1.8 zettabytes of data were generated worldwide in the year 2011 (Federal Big Data Commission, 2012), and there is no reason to assume that the rate of data generation has not continued to increase since then. Our current capacity to collect this data surpasses our capacity to extract value from it, and this will not change unless we can develop more intelligent KDD tools capable of shouldering some of the responsibilities traditionally reserved for human KDD experts.

The other trend we need to consider is the application of KDD technology by non-expert individuals. Concepts such as lifelogging and the quantified self (Fawcett, 2016) have so far had relatively little impact outside of academic discourse, but there is now a growing market for inexpensive wearable physiological sensor devices and associated software capable of generating reasonably accurate and meaningful feedback on aspects of well-being such as physical activity and sleep; an evaluation of a number of currently available devices and apps is provided in (Wen et al., 2017). Using such software does not require any particular expertise, but this comes at the cost of strict limits on its functionality. If, however, the software could help the user find solutions to more complex problems using more varied inputs, there would be potential here for fundamental societal change in, for instance, how healthcare is organized.

From the perspective of the KDD process, the essential question here is the division of labor between human and machine in the process. The seminal KDD process model of (Fayyad et al., 1996) does not, in fact, explicitly address the question of who is responsible for carrying out the tasks prescribed by the model, but among the models proposed later by other authors there are some that provide a more detailed account of the role of human actors in the process. These include the models presented in (Brachman and Anand, 1996), (Büchner et al., 1999) and (Moyle and Jorge, 2001), and also the standard process model CRISP-DM (Wirth and Hipp, 2000). However, a KDD process model that treats technology as an actor rather than a passive tool was first proposed in (Tuovinen, 2014) and then expanded upon in (Tuovinen, 2016). This model also offers some tentative ideas on the nature of knowledge in KDD and how knowledge should be represented in order to enable KDD software to process it in more advanced ways. Still, it appears that currently there is no model of the KDD process that provides a detailed account of how the evaluation/interpretation phase should be executed.

Other work on knowledge representation in the KDD process is also of interest here; particularly on the representation and application of domain knowledge in KDD there is a substantial body of previous research (Cao, 2010; Aslam et al., 2014; Dou et al., 2015). In more theoretical research, the formal concept and concept lattice formalisms associated with formal concept analysis have attracted considerable attention (Kuznetsov and Poelmans, 2013; Poelmans et al., 2013a; Poelmans et al., 2013b). Semantic Web technologies can also be applied in the context of KDD (Ristoski and Paulheim, 2016). Processing domain knowledge is usually seen as a way of improving the quality of KDD results, but it is argued in this paper that the relevance of knowledge representation to KDD is not limited to this; instead, any effort

to automate the evaluation and interpretation tasks of the KDD process must be founded on an appropriate knowledge representation scheme.

An even more interesting research topic is the representation of knowledge concerning the KDD process itself. The results of this work are particularly relevant because they can, at least in theory, be used to (partially) automate the generation of KDD workflows. Research in this area has been done since at least as early as 2005 (Bernstein et al., 2005), but the topic has begun to attract more attention during the current decade. Besides proposals aimed specifically at automation of workflow generation such as those presented in (Žáková et al., 2011) and (Keet et al., 2015), there have also been more theoretical efforts seeking to create a comprehensive ontology of KDD concepts, the OntoDM ontology (Panov et al., 2014) being perhaps the most prominent among these.

A 2013 survey (Serban et al., 2013) reviews research on what the authors call intelligent discovery assistants (IDA), by which they mean essentially the same thing as what in this paper are referred to as software KDD experts. As one would expect, the review shows a clear correlation between the number of data analysis tasks an IDA supports and the level of expertise it requires from its user: IDAs suitable for inexperienced users are limited to a relatively small number of tasks, whereas those that provide comprehensive support for the construction of KDD workflows can only be used effectively by an expert user. While there have been some notable efforts in the few years that have passed since the survey, (Nguyen et al., 2014) and (Kietz et al., 2014) being two examples, the situation has not changed radically. However, the next major advance may come in the not-too-distant future; the remainder of the paper explores the question of what the next steps are and how they can be achieved.

3 DECONSTRUCTING THE DEFINITION OF KDD

At this point, let us remind ourselves once more what it is that KDD searches for: *patterns in data* that are *valid*, *novel*, potentially *useful*, and ultimately *understandable*. The Fayyad et al. definition thus gives us five individual terms that characterize knowledge as the concept is understood in KDD. Below, we examine each of these terms in turn in order to see what their contribution to the definition of knowledge is.

Patterns in Data: This is the most fundamental of the five characterizations: it specifies the type of knowledge KDD searches for. First, the knowledge originates from data, that is, *observations* of the real world, and the way these observations are made and recorded affects every subsequent step of the KDD process. For instance, any issues with the quality of the observations will be carried through the process and affect the quality of the resulting knowledge, unless detected and actively dealt with at a sufficiently early stage. Second, the knowledge comes in the form of patterns, that is, *regularities* in the distribution of the data that can be used to generalize the observations. Ultimately, the objective is to derive from the regularities a model that accurately represents the phenomenon being studied also in those parts of the input space that are not covered by the data.

Valid: This property specifies that the patterns discovered should represent an *actual phenomenon* rather than, for instance, random noise or observation error. In other words, the apparent regularities present in the input dataset should also be present in the real world such that if more data is collected from the same source, the same patterns will continue to show. If the patterns only pertain to a specific dataset, they are only marginally more interesting than the individual observations that comprise the dataset; in order to be candidates for new knowledge, the patterns need to be generalizable beyond the original data in which they were found. If, for example, the objective is to use the patterns to predict future observations – as it typically is – the predictions will not be reliable if they are based on patterns that generally do not occur but just happen to be present in the input dataset.

Novel: This property specifies that the patterns should provide us with *new information* about the phenomenon that generated the input data. To judge the novelty of the patterns, some contextual information is necessary. The KDD process begins with the formulation of a problem, and the exploration of the input dataset by means of data mining is expected to yield a solution to that problem. When designing an approach to the problem, a number of factors can be varied: the specific input variables used, the data mining algorithms applied and the values chosen for the parameters of the algorithms. If this overall configuration is one that is not known to have been tried before, then the information learned by running it on the input dataset will be novel.

Useful: This property is often referred to as the actionability of knowledge acquired via KDD: the technology is generally not applied out of sheer curiosity, but to learn something that can be used as a basis for making informed decisions. The utility of the patterns discovered obviously depends on their validity, but validity alone is not enough: patterns that are valid are not useful unless they satisfy the particular requirements of the *application* for which they are in-

tended. For example, if a predictive model is discovered that is clearly rooted in a real phenomenon but not sufficiently accurate for the specific application in which it is meant to be used, then it is, in a certain sense, valid but not useful. Furthermore, the utility of the model also depends on whether the costs of implementing and deploying it in a production environment are justifiable, given the gains.

Understandable: This property implies that simply finding a model that appears to be accurate is not satisfactory: there should be a plausible (and testable) *explanation* for why it works. In the absence of a satisfactory explanation, the model remains conceptually detached from the real world, and a good explanation also helps considerably in determining the limits of the applicability of the model. In fact, for many practical purposes we may substitute a description of the circumstances under which the model can be successfully applied for a more in-depth understanding, thus avoiding the much more difficult task of providing a cause-and-effect explanation for the model. We can therefore say that a model is understood by a software expert if the expert has all the information it needs in order to be able to apply the model autonomously.

We have now established an interpretation for each of the five properties of knowledge in KDD; they are pragmatic interpretations, but this is justified, given that KDD is an inherently pragmatic form of discovery, concerning itself with practical value rather than epistemological truth. Next, we proceed to examine the interdependencies among the four conditions that patterns discovered in data must satisfy: validity, novelty, utility and understandability. Based on these, we can observe that the most natural sequence for evaluating the conditions is the following:

1. *Novelty:* As discussed above, the novelty of a proposed KDD solution can be evaluated by comparing its configuration – data, algorithms and parameters – with those of previously tried solutions to the same problem. Since this can be done before the solution is executed, novelty is the first condition to be evaluated.
2. *Validity:* Evaluating the validity of the outcome depends on being able to quantify the strength of the supporting evidence. The required validation scores are available once the solution has been executed, so validity is the second condition to be evaluated.
3. *Understandability:* For the basic level of understanding defined above, details of the problem and the solution are required. Additionally, information regarding validation coverage is needed in order to be able to determine constraints within which the solution can be expected to produce good results. Therefore understandability can be evaluated once validity has been established.
4. *Utility:* The utility of a KDD solution depends on whether its applicability and performance are adequate for the purpose for which it is intended to be used. Since this information is available only after it has been established that all the other conditions are satisfied, utility is the final condition to be evaluated.

In the next section we transform this sequence into a series of concrete tasks that a software KDD expert needs to perform in order to satisfy the conditions.

4 THE KNOWLEDGE CONSOLIDATION PROCESS

To get started, we can divide the knowledge consolidation process into four major steps, corresponding to the four conditions that a KDD result needs to satisfy in order for the KDD effort to be considered a success. The steps are:

1. *Capture:* identify the result to be consolidated and prepare it for further processing
2. *Verify:* confirm that the result has sufficient support to be considered reliable
3. *Delineate:* identify constraints on the applicability of the result
4. *Integrate:* deploy the result as a component of a software system

Below, we look at each of these steps in turn to achieve a more detailed description of what needs to be done during each step and how the process advances from one step to the next.

Capture: The first step of the process is to prepare the new KDD result for consolidation, or to *capture* it in a format that allows it to be processed further. Apart from converting the result into a suitable representation, this involves supplementing it with metadata that will be required at later stages of consolidation in order to be able to evaluate the knowledge conditions. Immediately available as an outcome of the knowledge discovery process are descriptions of the problem and the solution, but at each subsequent process step, additional metadata is generated from data and metadata available at that point. Furthermore, a knowledge base containing the accumulated body of domain knowledge needs to be made accessible such that supplementary information can be retrieved from it when needed for decision-making.

Verify: In the previous section, we established that for practical purposes, we can take the novelty of a KDD result as implicitly guaranteed by the specification of the problem and the solution. Therefore the first condition that needs to be evaluated as part of the consolidation process is the validity of the result; in other words, the next process step is to *verify* the result. To do this autonomously, a software KDD expert needs to make a number of decisions concerning the selection of validation data and thresholds appropriate for the situation, for which it needs to pull relevant knowledge out of the domain knowledge base. New metadata is then generated to describe the proceedings of the verification step: the validation data, methods and parameters used and the results achieved.

Delineate: The next condition to be evaluated is understandability, which we defined above as the ability to know the circumstances under which a KDD result is applicable. For a software KDD expert to be able to make this decision, metadata is again required. Available at this point are the details of the problem, the solution and the verification; the expert needs to analyze these in order to *delineate* the subset of all possible input vectors for which both the validity of the result and the strength of the supporting evidence are high enough to justify treating the result as reliable. To know what is high enough, the expert needs to look up relevant information in the domain knowledge base. New metadata is generated to describe constraints on the applicability of the result.

Integrate: The final condition to be evaluated is the utility of the result, the ultimate objective being to *integrate* it into a software system capable of applying it autonomously. For integration to be possible, the result must be adequate for the task in every way: it must not suffer from constraints that would make it unsuitable, and its secondary quality characteristics such as memory consumption and running time must also be acceptable. Additional analysis and additional domain knowledge are required in order to carry out this final step; once the utility of the result has been established, it can be deployed by the software expert as a component of the system in which it is to be used.

Figure 1 illustrates the knowledge consolidation process as outlined above. Each step of the process follows a cycle: access the domain knowledge base for information relevant to the task at hand, use the information to support analysis of the data and metadata received as input, generate additional metadata based on the outcome of the analysis, and deliver all of the accumulated data and metadata as input to the next process step. Processing metadata in various ways is thus a defining aspect of what a software KDD expert needs to be able to do; we shall cover this topic in some more detail in the next section.

With the integration step successfully completed, the new KDD result has been consolidated to such a degree that a computer can use it as a solution to the original problem without human supervision. Let us now consider the other major part of the KDD process that currently needs to be carried out by human experts: setting up the KDD workflow. For a software expert to be able to do this, it needs, firstly, to understand how the KDD process works; this, as discussed in Section 2, is the topic of some interesting research concerning KDD ontologies. Secondly, it needs the ability to do more advanced reasoning with domain knowledge, enabling it to understand how to fill gaps in established knowledge using KDD. Therefore, to enable a software KDD expert to *extend* consolidated knowledge, we need to make a transition from the pragmatic knowledge we have been working with so far to a more advanced, conceptual type of knowledge. Regarding how such a transition can be made, we can make the following observations:

1. Utility is a fundamentally pragmatic concept and does not need to be given a conceptual intepretation. Therefore we can take the utility of a consolidated piece of knowledge as fully established.
2. Understandability can be interpreted as the ability to place a consolidated piece of knowledge in its proper context by identifying semantic connections between it and previously established knowledge.
3. Validity can be interpreted as the presence of corroborating established knowledge and the absence of unresolved conflicts with established knowledge; it can be evaluated once understandability has been established.
4. Novelty can be interpreted as the consolidated piece of knowledge representing a genuinely new addition to established knowledge; it can be evaluated once understandability and validity have been established.

In other words, the sequence in which the conditions are processed is now reversed. At the end, the body of established domain knowledge is updated with the new addition, which thus becomes part of the background knowledge for future processes of knowledge discovery and consolidation. Figure 2 illustrates this.

We now have a process model describing the tasks that a software KDD expert needs to carry out in order to consolidate a piece of knowledge. In the next section, we adopt a different perspective and examine the capacities that the software expert needs to have in order to be able to perform those tasks.

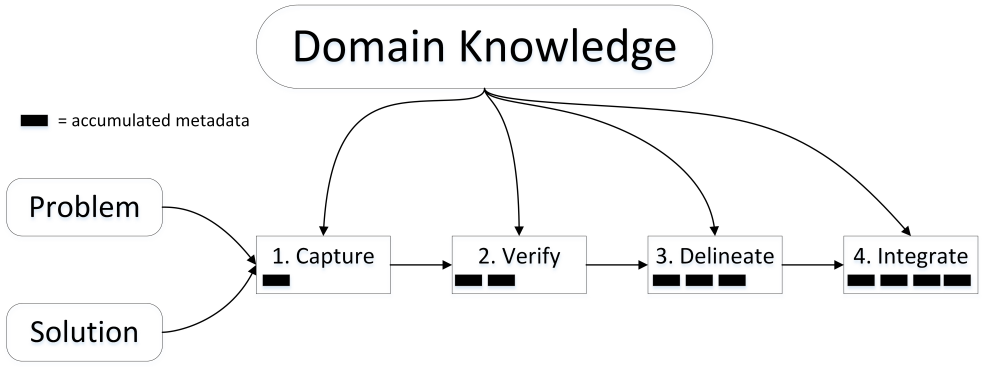

Figure 1: The main steps of the knowledge consolidation process and the accumulation of metadata over the course of the process. Additional knowledge is retrieved from the domain knowledge base at each step to enable the software KDD expert to make the decisions required in order to proceed to the next step.

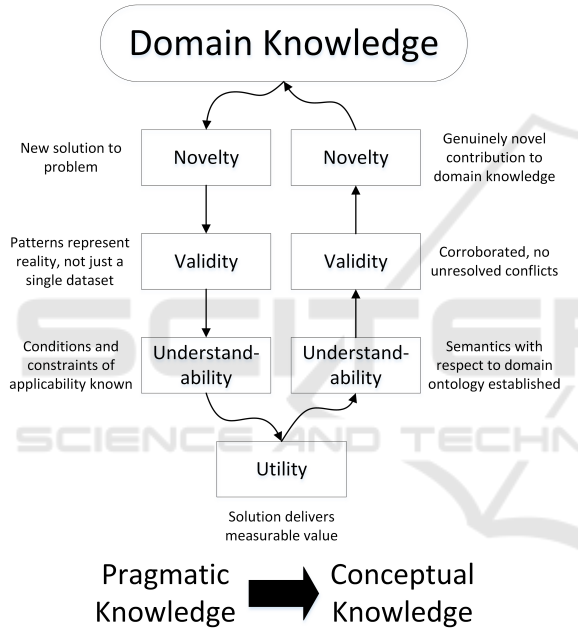

Figure 2: Knowledge consolidation and the transition from pragmatic knowledge to conceptual knowledge. The accumulated body of established domain knowledge represents both the background knowledge for new KDD efforts and the repository where the results of successful efforts are ultimately inserted.

5 THE CAPACITIES OF A SOFTWARE EXPERT

As observed above, the tasks of the knowledge consolidation process all involve the generation and manipulation of various kinds of metadata in various ways. We can therefore approach the required capacities of a software KDD expert from the perspective of metadata processing. As a result, four crucial capacities are identified:

1. *Representation:* maintaining the metadata in a suitable format
2. *Annotation:* generating metadata that conforms to the format
3. *Decision-making:* evaluating the knowledge conditions based on the metadata
4. *Reasoning:* processing the metadata on a semantic level

The capacities are in ascending order of complexity, with each capacity depending on all of the more basic capacities below it. Thus the first capacity that needs to be realized is representation, followed by annotation, decision-making, and finally reasoning. Below, each of these capacities is described in more detail.

Representation: As described above, each step of the knowledge consolidation process generates additional metadata, to be used in subsequent steps. To manage all of this diverse accumulated metadata, a suitable representation format and data structure is necessary; when new metadata is generated, it is inserted into the data structure, and when a step has been completed, the entire data structure is passed on as input to the next step. The format and structure need to be highly flexible in order to be able to accommodate all the different types of metadata generated over the course of the process, including the semantic information required when making the transition from pragmatic to conceptual knowledge.

Annotation: The capacity to represent metadata is a crucial enabler for other capacities, but it is an entirely passive capacity: it does not, in itself, allow a software expert to perform any operations on metadata. The annotation capacity is the most basic active capacity, enabling the software expert to write into the metadata structure as well as read from it. In other

words, a software expert with this capacity can generate new metadata and format it according to the metadata representation format so that it can be inserted into the metadata structure. However, the range of tasks that the software expert can carry out with this capacity alone is still very limited.

Decision-making: The annotation capacity gives a software expert the ability to generate information needed in the evaluation of the knowledge conditions, but does not enable it to use this information without being instructed by a human expert. With the decision-making capacity, the software expert can autonomously process the information in order to decide whether the KDD result being consolidated is sufficiently strong with respect to a given criterion. This capacity is thus what enables the software expert to apply the more basic representation and annotation capacities in a coordinated manner to walk through the steps of the knowledge consolidation process.

Reasoning: To work with conceptual knowledge, a software expert needs analytic capabilities that enable it to process semantic information representing the meaning of a KDD result with respect to the real world. With the reasoning capacity, the software expert is able to identify semantic links between the result and relevant domain knowledge; furthermore, it is able to analyze the implications of those links, allowing it to perform the final consolidation tasks left before the new KDD result can be inserted into the domain knowledge base as a fully established item of domain knowledge.

The question now is, what is required for us to be able to implement these capacities, and to what extent those requirements are already satisfied by existing work. From the review of related work in Section 2, we can see that there is already a considerable body of potentially useful research on knowledge representation, addressing both representation of the KDD process in general and representation of domain knowledge specific to individual KDD efforts. A synthesis of these should be enough to achieve at least a reasonable approximation of the representation capacity as envisioned here, although it is likely that some new work will still be necessary in order to fully realize it. From there, the leap to the annotation capacity would be comparatively small, and this would already enable the software expert to provide decision support for human KDD experts by generating additional information relevant to the decisions they need to make.

The decision-making capacity is a problem of a different order, requiring a level of intelligence where, instead of just generating information at the various stages of knowledge discovery and consolidation, the software expert is able to assess the significance of the generated information with respect to objectives, a crucial part of which is determining what the relevant objectives are in the first place. Thus, whereas a software expert with the annotation capacity would be able to generate metadata to support the evaluation of a given knowledge condition, but would leave the actual evaluation to be done by a human expert, a software expert with the decision-making capacity would be able to determine and execute the appropriate tests itself. Domain knowledge plays a critical part here, whereas the annotation capacity does not depend on external knowledge.

From decision-making to reasoning there is another major leap, the distinction being that whereas a software expert with the decision-making capacity can look up knowledge in the domain knowledge base and apply it while processing a KDD result, a software expert with the reasoning capacity is able to analyze the domain knowledge itself and evaluate the properties of the KDD result being consolidated on a conceptual or semantic level. In the broadest interpretation, this includes the ability to reason about cause-and-effect relationships in order to determine a satisfactory explanation for the result; it may take decades before artificial intelligence of this caliber is widely available to KDD practitioners. In the next section, we consider the question of what we may hope to accomplish in the meantime.

6 DISCUSSION

In its later stages, the knowledge consolidation process proposed in this paper begins to touch the realm of philosophy, and it is only fair to ask whether this is really necessary. KDD is, after all, fundamentally a data-driven form of discovery that concerns itself with statistical probability rather than incontrovertible truth. However, if we want an expert that has not just autonomy but *initiative* – the ability to decide what would be the most relevant and interesting KDD experiment to run next – we need to step beyond the basic premise of KDD and consider the significance of KDD results in terms of more than just their quantifiable instrumental value.

Our knowledge consolidation process is defined largely in terms of knowledge representation and metadata transformations, and this idea can be extended to cover the knowledge discovery process as well. The fundamental change implied here is that semantic metadata should be a seamless part of the KDD workflow from start to finish. The metadata encodes information about the data that enables the execution of knowledge consolidation tasks, and every

KDD operator used to transform the data must also transform the associated metadata in order to facilitate the knowledge consolidation process. This, as discussed in the previous section, would already be useful as a means of providing better decision support; the main requirements are the abilities to represent and generate the metadata, both of which are achievable in the short term. In fact, currently existing scientific workflow management systems already do something similar in that they generate provenance metadata to accompany experiment results; in (Esteves et al., 2016), a methodology for automating metadata generation in a more universal machine learning context is proposed.

In the medium term, the major problem to be tackled is how to build up the domain knowledge base to such a level that the software expert can make independent decisions instead of just helping human experts make them. A part of this problem is designing a suitable representation for the required KDD expertise, but by far the biggest challenge is actually obtaining the expertise so that it can be converted into the chosen representation and inserted into the knowledge base. From past experience, it seems unlikely that all this knowledge would be codified by hand; a much more promising possibility is for the software expert to learn it by observing the decisions made by human experts. This is an artificial intelligence problem, in the solution of which the concepts of meta-mining (Nguyen et al., 2014) and meta-learning (Lemke et al., 2015) are likely to prove relevant.

The impact of artificial intelligence technology on the practice of KDD is inevitably limited by the hardware requirements of running a powerful AI. The potential for a real revolution lies in making the technology available to ordinary individuals using regular computers, possibly even mobile devices. If the history of computing gives any indication of its future, the power of today's supercomputers will eventually be packaged in devices that are compact and cheap enough to be available to ordinary consumers. Even before that, however, there are at least two ways in which they may begin using KDD for their own purposes. One of these is off-the-shelf software designed to operate in a narrowly defined application domain with a restricted set of possible inputs and outputs, sacrificing universal applicability for higher utility under a constrained set of circumstances. Another possibility, and perhaps one with more interesting implications, is that instead of the user hosting the software expert on their local computer, they access it as a service running on a remote host. The software expert could then be of a more universal type, offering the user a lot more freedom in determining what data they wish to use and what kind of knowledge they are interested in extracting from it.

We can thus measure the completeness of a software KDD expert along a number of dimensions: how many steps of the knowledge consolidation process it can cover, how advanced its operational capacities are, and how freely it can be deployed in different application domains. A software expert that is completely functional with respect to all three dimensions is a distant vision, but as we have already established, a software expert can be useful without being complete. Usefulness is, in fact, probably what will determine the direction of future research and development: the emphasis will be on improving those dimensions where advances of practical importance can be achieved relatively quickly. Arguably the highest potential for the next major breakthrough lies in developing a software expert capable of autonomous decision-making in a domain such as the life sciences, where there is already a substantial amount of codified domain knowledge available in the form of ontologies such as the SIO (Dumontier et al., 2014), which is specifically intended by its authors to facilitate biomedical knowledge discovery.

7 CONCLUSION

In this paper, we discussed the concept of a software KDD expert: a piece of software capable of performing tasks that, at present, can only be handled by a human KDD expert. In this discussion, we focused on the task of knowledge consolidation: evaluating the result of the KDD process in order to confirm that has all the properties that a piece of knowledge is expected to have. By deconstructing the classical definition of KDD and examining interdependencies among its components, we derived a process model for knowledge consolidation in KDD. Furthermore, we discussed the specific capacities that a software KDD expert needs to have in order to carry out the tasks of the knowledge consolidation process, and reviewed the results of relevant research in order to see how far away we are from having the technology that would enable us to implement a software expert capable of executing at least one major knowledge consolidation task. We found that a software expert capable of providing decision support for a human expert is within reach, but a software expert capable of making decisions without the supervision of a human expert requires further advances in the fields of knowledge representation and artificial intelligence. The problem can be simplified by constraining the area of applicability of the software expert, so the decision-making

capacity is likely to emerge first in a narrowly defined application domain.

REFERENCES

Aslam, M., Talib, R., and Majeed, H. (2014). A review on the role of domain driven data mining. *International Journal of Computer Science and Mobile Computing*, 3(5):708–712.

Bernstein, A., Provost, F., and Hill, S. (2005). Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):503–518.

Brachman, R. J. and Anand, T. (1996). The process of knowledge discovery in databases. In *Advances in Knowledge Discovery and Data Mining*, pages 37–57. American Association for Artificial Intelligence.

Büchner, A. G., Mulvenna, M. D., Anand, S. S., and Hughes, J. G. (1999). An internet-enabled knowledge discovery process. In *Proceedings of the 9th International Database Conference*, pages 13–27.

Cao, L. (2010). Domain-driven data mining: Challenges and prospects. *IEEE Transactions on Knowledge and Data Engineering*, 22(6):755–769.

Dou, D., Wang, H., and Liu, H. (2015). Semantic data mining: A survey of ontology-based approaches. In *Proceedings of the 9th IEEE International Conference on Semantic Computing*, pages 244–251.

Dumontier, M., Baker, C. J. O., Baran, J., Callahan, A., Chepelev, L., Cruz-Toledo, J., Del Rio, N. R., Duck, G., Furlong, L. I., Keath, N., Klassen, D., McCusker, J. P., Queralt-Rosinach, N., Samwald, M., Villanueva-Rosales, N., Wilkinson, M. D., and Hoehndorf, R. (2014). The Semanticscience Integrated Ontology (SIO) for biomedical research and knowledge discovery. *Journal of Biomedical Semantics*, 5(1):14.

Esteves, D., Mendes, P. N., Moussallem, D., Duarte, J. C., Zaveri, A., and Lehmann, J. (2016). MEX interfaces: Automating machine learning metadata generation. In *Proceedings of the 12th International Conference on Semantic Systems*, pages 17–24.

Fawcett, T. (2016). Mining the quantified self: Personal knowledge discovery as a challenge for data science. *Big Data*, 3(4):249–266.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34.

Federal Big Data Commission (2012). Demystifying big data: A practical guide to transforming the business of government. Technical report, TechAmerica Foundation.

Keet, C. M., Ławrynowicz, A., d'Amato, C., Kalousis, A., Nguyen, P., Palma, R., Stevens, R., and Hilario, M. (2015). The Data Mining OPtimization Ontology. *Web Semantics: Science, Services and Agents on the World Wide Web*, 32:43–53.

Kietz, J.-U., Serban, F., Fischer, S., and Bernstein, A. (2014). "Semantics inside!" but let's not tell the data miners: Intelligent support for data mining. In *The Semantic Web: Trends and Challenges. 11th International Conference, ESWC 2014, Proceedings*, pages 706–720.

Kuznetsov, S. O. and Poelmans, J. (2013). Knowledge representation and processing with formal concept analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(3):200–215.

Lemke, C., Budka, M., and Gabrys, B. (2015). Metalearning: a survey of trends and technologies. *Artificial Intelligence Review*, 44(1):117–130.

Moyle, S. and Jorge, A. (2001). RAMSYS - a methodology for supporting rapid remote collaborative data mining projects. In *ECML/PKKD01 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, pages 20–31.

Nguyen, P., Hilario, M., and Kalousis, A. (2014). Using meta-mining to support data mining workflow planning and optimization. *Journal of Artificial Intelligence Research*, 51:605–644.

Panov, P., Soldatova, L., and Džeroski, S. (2014). Ontology of core data mining entities. *Data Mining and Knowledge Discovery*, 28(5–6):1222–1265.

Poelmans, J., Ignatov, D. I., Kuznetsov, S. O., and Dedene, G. (2013a). Formal concept analysis in knowledge processing: A survey on applications. *Expert Systems with Applications*, 40(16):6538–6560.

Poelmans, J., Kuznetsov, S. O., Ignatov, D. I., and Dedene, G. (2013b). Formal concept analysis in knowledge processing: A survey on models and techniques. *Expert Systems with Applications*, 4(16):6601–6623.

Ristoski, P. and Paulheim, H. (2016). Semantic Web in data mining and knowledge discovery: A comprehensive survey. *Web Semantics: Science, Services and Agents on the World Wide Web*, 36:1–22.

Serban, F., Vanschoren, J., Kietz, J.-U., and Bernstein, A. (2013). A survey of intelligent assistants for data analysis. *ACM Computing Surveys*, 45(3):article 31.

Tuovinen, L. (2014). *From machine learning to learning with machines: Remodeling the knowledge discovery process*. PhD thesis, University of Oulu, Finland.

Tuovinen, L. (2016). A conceptual model of actors and interactions for the knowledge discovery process. In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management – Volume 1: KDIR*, pages 240–248.

Žáková, M., Křemen, P., Železný, F., and Lavrač, N. (2011). Automating knowledge discovery workflow composition through ontology-based planning. *IEEE Transactions on Automation Science and Engineering*, 8(2):253–264.

Wen, D., Zhang, X., Liu, X., and Lei, J. (2017). Evaluating the consistency of current mainstream wearable devices in health monitoring: A comparison under free-living conditions. *Journal of Medical Internet Research*, 19(3):e68.

Wirth, R. and Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, pages 29–39.