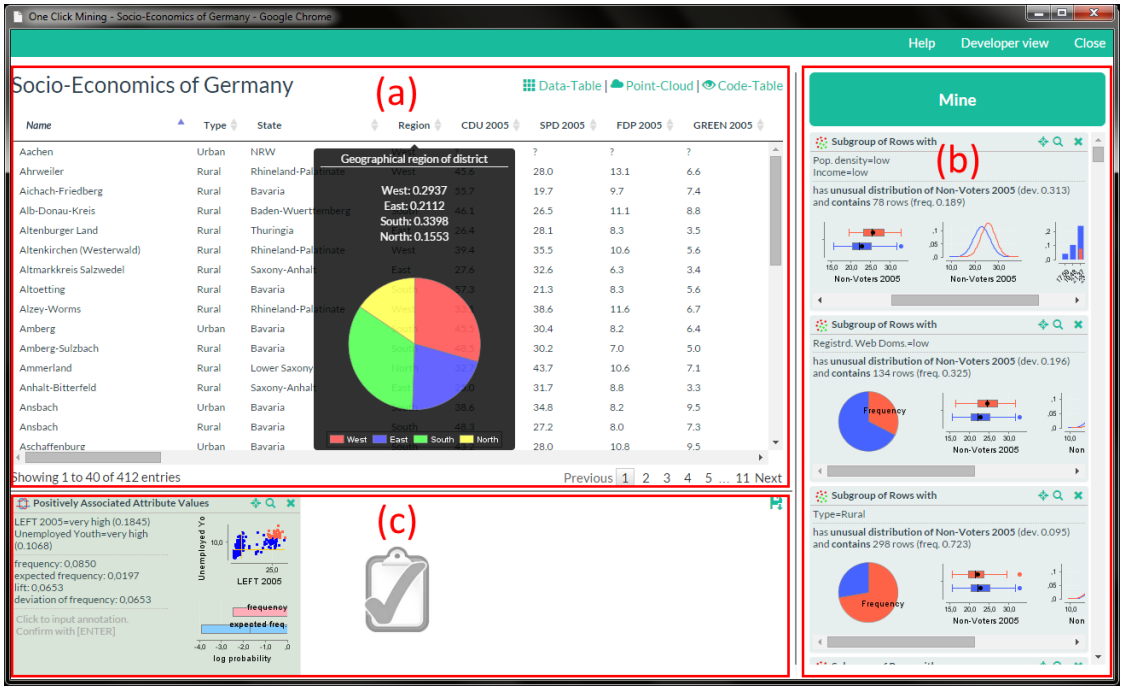

Figure 3: Creedo analytics dashboard with data view container (a), pattern mining launcher (b), and KD result container (c).

embedding of the data, and a propositional logic view, which renders a collection of propositional statements defined on rows of a data table (as required by pattern discovery algorithms that work on binary data).

The component that allows to perform the actual data mining is referred to as the **pattern mining launcher**. It enables the user to run pattern discovery algorithms on all data-artifacts present in the system and to display their results. The set of available algorithms can be chosen by the system designer from all realKD mining algorithms. In order to launch one of the algorithms, the user first has to click on a mine button, then choose the specific algorithm she wants to use, and finally select valid settings for all the parameters of that algorithm (the last two steps are only required when more then one algorithm is available and the selected algorithm exposes user parameters, both of which can be avoided by the system designer in order to provide a non-expert user-experience). After the termination of the algorithm, the component displays all patterns found in a scrollable list below the mine button. This list is referred to as the **candidate area** because patterns *can* be taken over from here into the user's result set should she consider them suitable. For the purpose of defining study designs, the accessible state of the pattern mining launcher includes, in addition to the content of the candidate area, also the algorithmic settings that have been used by the user to find the patterns in it.

Finally, the **knowledge discovery result container** allows users to incrementally assemble result sets for their data analysis task. Results are received by means of drag-and-drop from the candidate area of the pattern mining launcher. They also can be deleted, re-arranged dynamically, e.g., in order to express a (linear) result prioritization, and annotated in order to express a result interpretation. The accessible state of this component is the ordered set of results along with all annotations as well as the time and number of mining rounds that have passed until each pattern was found.

In summary, the components of analytics dashboards are all designed towards simplicity and a strictly defined user-purpose. This serves two goals: the resulting data analysis systems are relatively accessible also to inexperienced users and the components allow for an unambiguous interpretation of user interaction—both of which are useful properties for test systems in user studies.

4. STUDY DESIGN COMPONENTS

After having an overview over the possible user-experiences that can be provided by the Creedo web application, we are now ready to turn to the more abstract aspects of study definitions. As discussed earlier, the central notion of Creedo's study domain language is the study design. In this section we now present in more detail the individual components of those designs: system and task specifications, performance metrics, and assignment schemes.

4.1 System and Task Specifications

A **system specification** within a study design is a specification of how to construct an analysis system given certain input parameters. When using the Creedo web application, this means how to construct an analytics dashboard from the components presented in Sec. 3. More generally, system specifications have the role of determining all aspects of the user-experience that are relevant to the study hypothesis. Typically, this means that different specifications within the same study design are identical except for a specific test component, the effect of which is to be determined by an

extrinsic evaluation. For example, the test system might contain a specific data visualization (e.g., PCA-based point cloud) whereas the control system does not, or the test system replaces specific algorithm parameters (e.g., support/lift in association mining) by an automatic choice whereas the control system exposes those parameters to the user.

The input parameters of the system specifications are then used during study execution to inject task specific content provided by the task specifications into the analysis systems. In the Creedo web application, the current default implementation of a system specification accepts a collection of **data loaders** that determine the data artifacts available in the dashboard and a collection of **component builders** that shape its visual appearance.

While system specifications define tools, **task specifications** describe the purpose and evaluation context, in which these tools are supposed to be used within a study. Looking back to our hypothesis blueprint (1), we are interested in assessing how well human users can solve specific tasks with certain analysis systems. That is, we are interested in how well humans can operate a software-tool with the purpose of producing results (a solution) for some task.

Naturally, this purpose has to be communicated to the user in a human-interpretable form. At the same time, the data analysis system expects a machine-readable input. Then, the evaluation of the hypothesis requires actual solution entities, for which certain qualities can be assessed. Hence, the state of the tool, in which the participant considers the task as solved, has to be interpreted as one or more of these result entities. Finally, also the human evaluators, if they are involved, need to have an understanding of the task in order to carry out their evaluations. Thus, another set of human-understandable instructions, yet from a different perspective, is required in studies which involve evaluators. In summary, task specifications in Creedo essentially have to be able to translate an abstract concept of a task between several ontological realms and between several perspectives. For the Creedo web application they consist of

1. human-understandable **instructions**, i.e., a sequence of HTML-documents that may refer to embedded media files,

2. **input values** of parameters accepted by all of the system specifications in the study design (e.g., the input data for which the task has to be carried out and other auxiliary input that is task specific),

3. a **result definition**, i.e., a rule that describes how to extract result entities from a state of the dashboard, in which the participant has declared that she is done working (this also includes determining whether a given state of the dashboard can be declared as "done" in the first place), and

4. a set of **evaluation schemes** that are used by human evaluators to evaluate task results.

At the moment, all evaluation schemes in Creedo are **rating schemes** that consist of a name, a set of natural language evaluation instructions, and a finite set $V \subseteq \mathbb{Z}$ referred to as the **scale** of the rating scheme. For a result r, a single evaluation according to such a rating scheme is then simply the value $v \in V$ that is assigned to r by one evaluator.

4.2 System Performance Metrics

Creedo formalizes the concept of one analysis system being "better" than another by comparing them through a system performance metric. These metrics are supposed to capture intuitive concepts like the "average quality of results produced (with a specific system)" or "average time needed by participants to produce a good result". The formal definition of such metrics for systems can involve other metrics defined on the different types of artifacts that are produced during study execution. That is, system metrics can rely on session metrics, session metrics on results metrics, and result metrics on evaluation metrics. On each of these levels, Creedo allows the definition of metrics according to certain production rules, but also provides a set of useful predefined metrics. Before we introduce the general language for metric definitions, let us consider some of the predefined metrics in order to get a sense for their general flavor.

Let c be an evaluation scheme given in the task specification of the study, and E_x denote the set of all c-ratings that have been performed for a result $x \in X$. A basic result metric is then the **average c-value** defined by $\hat{c}(x) = \operatorname{avg}\{c(e)\colon e \in E_x\}$. A slightly more complex variant of this metric abounds from applying a z-score transformation to all rating values assigned to results by a specific evaluator. This gives rise to the **average evaluator-normalized c-value** defined by $\hat{c}_*(x) = \operatorname{avg}_{e \in E_x}(c(e) - \mu_{u(e)})/\sigma_{u(e)}$ where $u(e)$ denotes the evaluator that has provided evaluation e and μ_u and σ_u denote the sample mean an standard deviation of all c-ratings provided by a particular evaluator u. This metric can be useful when the study designer suspects that some evaluators might interpret the rating scale of c differently than others. As an example for a full system performance metric consider the **average maximal f-value** that is defined for any result metric $f\colon X \to \mathbb{R}$ by

$$a \mapsto \operatorname{avg}\{\max_{x \in X_s} f(x)\colon s \in S_a, |X_s| > 0\}$$

where S_a denotes the set of all sessions that have been performed using system a, and X_s the set of all results that have been produced in session s. Another example is the **median time until success** that is again defined with respect to some result metric f and a success threshold $\tau \in R$ by

$$a \mapsto \operatorname{med}\{\min\{t(x)\colon x \in X_s, \hat{c}(x) > \tau\}\colon s \in S_a, |R_s| > 0\} \tag{2}$$

where $t(x)$ denotes the total session time until the result x was stored by the participant.

Creedo's general **language for metric definitions** is based on a set of elementary metrics (which can vary depending on context and implementation) and a set of production rules from which more advanced metrics can be defined. Let us first review some **elementary metrics**, which are defined in most contexts. For rating evaluations, elementary metrics contain:

- `Value`, i.e., the rating value chosen by the evaluator,
- `EvaluationTime`, i.e., the time taken by the evaluator to choose value.

For results, some elementary metrics are:

- `SessionTimeUntilStored`, i.e., the total time since the trial session started at the moment a result is added to the result area.

- `RoundsUntilFound`, i.e., how often the participant had to start a mining algorithm until the result was shown in the candidate area.
- `TimeBetweenFoundAndStored`, i.e., how long it took the participant to find and store the result after it has been produced by a mining algorithm.

The set of elementary session metrics includes:

- `TotalSessionTime`, i.e., the total time between the moment the participant first sees the analysis system until the system is closed,
- `RoundsInSession`, i.e., the total number of times the participant has pressed the mine button within the session,
- `NumberOfResults`, i.e., the number of results that have been submitted by the participant at the end of the session.

Now we can turn to the **production rules** for building more complex X-metrics, where X can be one of the sets of analysis systems, analysis sessions, results, or rating evaluations, i.e., $X \in \{A, S, R\} \cup \{E_c : c \in C\}$ (the rules also involves X-**constraints**, which are boolean function $q\colon X \to \{\text{true}, \text{false}\}$):

- If f and g are X-metrics then $x \mapsto f(x) + g(x)$, $x \mapsto f(x) - g(x)$, $x \mapsto f(x)g(x)$, $x \mapsto f(x)/g(x)$, and $x \mapsto f(x)^{g(x)}$ are also X-metrics.
- The trivial constraint $x \mapsto \text{true}$ is an X-constraint.
- If f is an X-metric then for all thresholds $\tau \in \mathbb{R}$ the functions $x \mapsto \delta(f(x) > \tau)$, $x \mapsto \delta(f(x) = \tau)$, and $x \mapsto \delta(f(x) < \tau)$ are X-constraints where δ evaluates to true if the condition given in brackets is true and false otherwise.
- If q and r are X-constraints then $x \mapsto q(x) \wedge r(x)$ and $x \mapsto \neg q(x)$ are also X-constraints.
- If g is a Y-metric, q is a Y-constraint, and $X \subseteq 2^Y$ then

$$x \mapsto \text{aggr}\{g(y)\colon y \in x, q(y)\}$$

  is an X-metric for all aggregation functions

$$\text{aggr} \in \{\max, \min, \text{avg}, \text{med}, \text{mode}, \text{count}\} \ .$$

Here, as a slight abuse of notation, we identify analysis systems with the set of analysis session that have been performed with them, i.e., $A \subseteq 2^S$, analysis sessions with the set of their results, $S \subseteq 2^X$, and result with the set of rating evaluations that they have received $X \subseteq 2^{E^c}$ for all rating schemes c in the task specification.

4.3 Assignment Schemes

The system performance metrics of a study design define a lot of measurements that can be taken based on the actions of study users (participants and evaluators). What these metrics do not determine is how an actual pool of study users should be assigned to perform these actions. This is the role of **assignment schemes**. In particular, they have to solve the following problems:

- *User workload has to be limited.* Each user has a limited amount of time and attention. If the study is demanding more of these resources than the user is willing to provide, she will not respond, and the user is essentially lost for the study.
- *Biases have to be controlled.* For instance, a participant working on the same task with two different systems, is likely to improve the second time independently of the system due to an increased understanding of the task itself. Another important source of biases abounds if users act in the double role of participants and evaluators: the study designer has to take into account how the knowledge that a participant gains when working on a task will effect her judgment when she is assigned to evaluate results of that task afterwards.
- *Dependencies of assignments have be resolved.* Trivially, evaluation tasks can only be assigned when the corresponding results have been produced in trials. On top of that, if a uniform distribution of evaluations is required for all results, then it makes sense to hold back evaluation assignments even further until the total number of results is known.

In summary, assignment schemes shape the behavior of a study at execution time by answering the following questions: Which participant is supposed to work with which system on which task, which evaluator is supposed to evaluate which result, and when, i.e., in what sequence, are these actions to be performed?

Currently, two schemes are available: `TwoPhaseAssignment`, which first issues all trial assignments to all participants and then, when those are all completed, issues all evaluation assignments, and `CombinedAssignment`, which generates a combined trial/evaluation assignment for users that have both roles. Both schemes have the following parameters in common:

- Number of trial assignments per participant.
- Constraints for valid trial assignments which can be a subset from
  - `IdenticalSystems`, i.e., all systems in all trial assignments for a given participant have to be identical,
  - `IdenticalTasks`, i.e., all tasks in all trial assignments for a given participant have to be identical,
  - `NoIdenticalTasks`, i.e., in no two tasks assignments of a given participant she is allowed to work on the same task,
  - `NoIdenticalSystems`, i.e., in no two tasks assignments of a given participant she is allowed to work with the same systems.
- Number of results per evaluator.
- Constraints for valid evaluation assignments which can be a subset from
  - `NoEvaluationOfOwnResults`, i.e., no result given for evaluation has been produced by the evaluator herself in one of her own trial assignments,

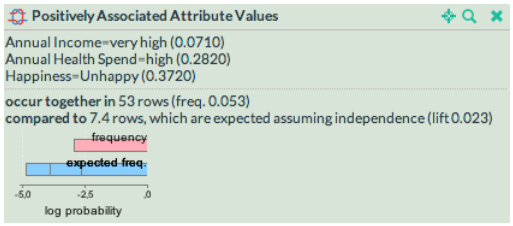

Figure 4: Association pattern stating that the joint frequency of three attribute/value combinations is "higher then expected". Intuitively, this statement is less interesting for someone who already knows that "annual health spend" is correlated with "annual income". FORSIED aims to quantify this effect.

- `NoEvaluationOfOwnTasks`, i.e., no result given for evaluation is of the same task as a task that the evaluator worked on herself,
- `ResultsStratifiedBySystems`. i.e., the results within an evaluation assignment have to come from all different systems at approximately equal amounts.

Both schemes assure that all system/task combinations will receive an approximately equal number of sessions and all results will receive an approximately equal number of evaluations.

5. USE CASE: EVALUATING FORSIED

In order to illustrate the various concepts of the Creedo framework that were introduced in the previous sections, we now turn to an exemplary case study. Namely, we present the design of a study that we conducted to evaluate the framework for "Formalizing Subjective Interestingness in Exploratory Data Mining" (FORSIED) [De Bie, 2013, De Bie and Spyropoulou, 2013]—a recently proposed pattern mining technique with a particularly user-centric approach. Note that a detailed discussion of this study and its results is out of scope of this paper, and will be published separately. Here we focus on the mapping of a real study hypothesis with theory-specific requirements to Creedo components.

5.1 Background and Hypothesis

FORSIED aims to quantify the interestingness of a pattern for a particular user depending on the prior knowledge that this user already has about the dataset, in which the pattern was found. Prior knowledge can come, e.g., in the form of certain summary statistics of the data, like row and column averages, or by other patterns that have been previously discovered. Importantly, FORSIED is an attempt to capture a universal notion of interestingness that in an ideal implementation (where all prior knowledge of a user can be assessed) coincides with the intuitive natural-language notion of interestingness. In order to make this very general concept more tangible, we focus here on an embodiment where we use a FORSIED-based pattern ranker as post-processing step in a round-based discovery of association patterns (see Webb [2010] and Fig. 4). This process works as follows: every round starts with the production of a random set of association patterns that, subsequently, is ordered by a ranker, before it is presented to the user. Then the user can pick from that ordered list patterns according to her interest and start over into the next round until she is done.

The FORSIED-based ranker, in particular, orders patterns according to their subjective interestingness considering as prior knowledge all the patterns that have already been discovered and stored by the user in previous rounds as well as the univariate distributions of all data attributes. Based on the design claim of FORSIED, in every round, the FORSIED-based ranker should point the user directly to those new patterns that are still interesting to her, and consequently allow her to save time and attention while browsing the result set compared to when a traditional static measure is used for ranking. The longer the discovery process proceeds the more this advantage should be emphasized. This gives rise to the following **hypothesis**:

> *"A FORSIED-based association discovery process allows users to discover a set of interesting patterns from a dataset faster than a conventional association discovery process (based on a static interestingness measure that is oblivious to prior and gained knowledge)."*

Translating this hypothesis into a useful operational study design implies defining a sufficiently robust objective measurement about subjective interestingness. In the next section, we will see how this apparent conundrum can be attacked by using Creedo components in order to control the knowledge of participants and evaluators throughout study execution.

5.2 Design Requirements

In addition to developing an executable study design that captures as precisely as possible our hypothesis, we also aim for a design that meets the following general requirements:

- It should evaluate the claim about "user interestingness" *extrinsically*, i.e., this notion should not just be captured by simplifying formal assumptions. In particular, this means that we want to employ human evaluators for assessing whether a result is interesting or not, instead of relying, e.g., on the intra-theoretic assumption that FORSIED's formalization of interestingness does indeed correspond to the intuitive notion thereof.
- Moreover, the design should be *robust*, i.e., the study result should be affected as little as possible by the potentially outlying behavior of individual participants or evaluators. This means that we want to average over different trials and result evaluations.
- Finally, we aim for a design that leads to *scalable* studies, i.e., the amount of time and attention required by the study owner should not depend on the number of participants. This means that we want to assign to users the double role of trial participant and result evaluator.

These three requirements have two practical implications for our study design. First, since we want to be able to meaningfully average over results, it must be possible for all task results to evaluate them on an identical scale. That

System	$\tau = 1$	$\tau = 2$	$\tau = 3$
a_{test}	**563.63**	**563.63**	**505.5**
a_{control}	694.25	803.96	inf

Table 1: Median time in seconds until success corresponding to the two systems and different success thresholds.

is, interestingness of a result should mean the same for all participants and evaluators. Thus, our task definitions (in particular instructions and dataset) have to control the prior knowledge among all users, because, as per our hypothesis, prior knowledge is what determines interestingness. Since we also want to put users into the double role of participants and evaluators, this creates the further difficulty that when a user is asked to evaluate a result, she must have the same prior knowledge at that moment as the participant when she created the result. Hence, we have to define two task variants such that performing one task as a participant does not change the prior knowledge for the other variant (and hence still allows an unbiased evaluation of it).

5.3 Study Design

The **system variants** for the study design are already more or less defined by the considerations in Section 5.1. In particular we have a_{test} corresponding to an analytics dashboard equipped with the FORSIED-based ranker and a_{control} corresponding to one with a conventional ranker based on the lift measure. As actual association discovery algorithm we fix a controlled pattern sampling algorithm [Boley et al., 2012] where all parameters (such as number of patterns per round, pattern pruning strategy) are fixed to feasible values. This makes the study design accessible also for participants that are not pattern discovery experts.

As stated above, we need two **task variants** in order to meet our requirements. In order to control the prior knowledge, the **input datasets** consist of randomly generated population data of two fictitious lands called "Lakeland" and "The Plain". Each dataset contains 1000 rows, each of which corresponding to a fictitious inhabitant that is sampled according to a joint probability distribution of the following variables:

- **Race** $\in$ {Griffin, Diricawl, Werewolf},
- **Region** $\in$ {east, west, north, east} (of residency),
- **Income** $\in \{0, 1, 2, \ldots, 1000\}$ (in gold coins),
- **Health spending** $\in [0, 1]$ (as fraction of annual income), and
- **Happiness** $\in$ {happy, unhappy}.

The probability distributions for each of the two lands have different multi-variate dependencies and model parameters in order to "inject" certain patterns into the datasets such as, e.g., $P[\textbf{Race} = \text{Diricawl}, \textbf{Region} = \text{north}]$ being much larger than the product $P[\textbf{Race} = \text{Diricawl}]P[\textbf{Region} = \text{north}]$. In addition to the input dataset the input values also contain the univariate statistics of all attributes for the FORSIED-based ranker. Corresponding to the two input data generators, the two variants of **task instructions** are:

> "You see a sample of the population of **(Lakeland/The Plain)**. Get familiar with the summary statistics of each attribute by [...]. Then click the mine button in order to find patterns in the population. You can save patterns you consider interesting by [...] and delete patterns by [...]. Repeat these steps to refine your pattern collection until you think you have discovered the most interesting collection consisting of exactly 3 patterns."

The **result definition** is then the collection of patterns present at the result container of the analytics dashboard considered as *a single result set* (and a result can only be submitted if there are exactly three patterns in this area). This corresponds to our hypothesis, in which we conjecture that the FORSIED-based system should have an advantage when considering the time it takes to construct a whole set of result patterns. In contrast, since the FORSIED-based system and the conventional system behave essentially identical before the first pattern is stored, we would not expect to see a substantial advantage on the single pattern level. Finally, this also has to be reflected in the **evaluation scheme** used for result evaluation. Here, we use a single elementary rating metric called "joint interestingness" with the instructions:

> "How interesting is this set of patterns as a whole (taking into account the elementary statistics of the data attributes)?"

and scale {0 (not), 1 (almost not), 2 (a little), 3 (somewhat), 4 (very)}.

Corresponding to our initial considerations, we choose `TwoPhaseAssignment` as **assignment scheme** with the parameter values of 1 for the number of trial assignments per participant, 3 for number of results per evaluator, and `NoEvaluationOfOwnTask` as constraint for the evaluation assignments. Using the constraint assures that evaluators will see results only from the task variant they did not work on themselves. Thus, as required, they will have the same prior knowledge when evaluating the task as the participant had when producing it.

As **system performance metrics** we can then define the median time until success (see Eq. (2)) for different thresholds, e.g., $\tau \in \{1, 2, 3\}$ corresponding to different requirements to the average ranking of a result to be considered a success. As a relatively strict criterion, we can then say that our hypothesis is supported by a study, if for all $\tau \in \{1, 2, 3\}$, we have $f_\tau(a_{\text{test}}) < f_\tau(a_{\text{control}})$ where f_τ denotes the system performance metric with respect to τ.

5.4 Results and Experiences

The authors conducted a study based on the above design with 16 participants/evaluators who followed an open invitation among our department members and master students. While this population was recruited from a friendly environment, note that based on the study setup, there was no reliable way for participants to tell with what system variant they were working or what system was used to produce the result they were evaluating. Table 1 contains the aggregated results of this study for the different strictness levels of the system performance metric. As we can see, it was indeed the case that for all three levels the median time until success was smaller for the test system than for the target system.

Using the Creedo web application, the whole study process could be administrated conveniently from one location, while participants and evaluators where able to fulfill their assignments from wherever they wanted within a previously fixed six day period (the first half of which was reserved for trial participation, the second half for result evaluation). Moreover, the study design is readily available for re-use with a different population of users—either unmodified in order to increase the confidence in our initial result or in a modified form in order to address specific doubts or to investigate refined follow-up questions.

6. SUMMARY AND OUTLOOK

As we have seen, Creedo can support researchers in defining and conducting repeatable studies with real users in order to extrinsically evaluate novel contributions in data mining. The focus of this support is at the moment on the easy definition of analysis systems, tasks, measurements, and assignment logic in order to control biases and to reduce the cost of the actual study organization and performance measurement. In particular, Creedo allows to employ participants also in the role of result evaluators in order to provide scalable study designs.

The most important limitations of Creedo's current state are perhaps as follows. While a relatively rich set of performance metrics can be expressed in the system, Creedo currently does not provide any support for the statistical interpretation of those metrics. That is, in case the study authors do know that their participants are a representative subset of a certain target demographic, there is no support for testing whether metric measurements are significant for that demographic. Moreover, reflecting current restrictions of the realKD library, the data sources that can be injected into analytics dashboards of the web application are limited to rather small-scale tabular data. Finally, the available visual mining components are somewhat specific to pattern discovery. However, as our survey among ECMLPKDD authors revealed, there appears to be a high demand for user studies also in other subfields of data mining, especially of course in visual analytics. Hence, extending the components available towards these areas is another major direction of growth.

Among the many directions for potential improvement, the authors believe that the extension of Creedo and its implementation should be mainly driven by the requirements emerging from actual practical attempts of extrinsically evaluating novel data mining techniques. If there is a developing culture in data mining research of performing studies with real users, then the community as a whole will understand better than today, what are central requirements for supporting this task. The organization of the Creedo web application as an open source project is a suitable basis for an organic growth that follows such a development. The authors are excited to engage in collaborations with other data mining research groups to jointly gain more experience in performing user studies and foster their growth.

Acknowledgment

This work was supported by the German Science Foundation (DFG) under the reference number 'GA 1615/2-1'.

References

Creedo web application. `https://bitbucket.org/realKD/creedo/`.

Kwiksurveys. URL `https://kwiksurveys.com/`.

realKD Java library. `https://bitbucket.org/realKD/realkd/`.

Surveymonkey. `https://www.surveymonkey.com`.

M. Boley, S. Moens, and T. Gärtner. Linear space direct pattern sampling using coupling from the past. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 69–77, 2012.

M. Boley, M. Mampaey, B. Kang, P. Tokmakov, and S. Wrobel. One click mining: Interactive local pattern discovery through implicit preference and performance learning. In *Proceedings of the ACM SIGKDD Workshop IDEA*, pages 27–35. ACM, 2013.

M. Boley, B. Kang, and T. Gaertner. Poll among ECML-PKDD authors on user studies. `http://www.realkd.org/dm-userstudies/ecmlpkdd-authorpoll-march2015/`, 2015. [Online; accessed 2-April-2015].

T. De Bie. Subjective interestingness in exploratory data mining. In *Advances in Intelligent Data Analysis XII - 12th International Symposium, IDA 2013, London, UK, October 17-19, 2013. Proceedings*, pages 19–31, 2013.

T. De Bie and E. Spyropoulou. A theoretical framework for exploratory data mining: Recent insights and challenges ahead. In *Machine Learning and Knowledge Discovery in Databases*, volume 8190 of *LNCS*, pages 612–616. Springer Berlin Heidelberg, 2013.

V. Dzyuba, M. v. Leeuwen, S. Nijssen, and L. De Raedt. Interactive learning of pattern rankings. *International Journal on Artificial Intelligence Tools*, 23(06), 2014.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3): 37, 1996.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

D. J. Hand. Pattern detection and discovery. In *ESF Exploratory Workshop on Pattern Detection and Discovery*, pages 1–12. Springer, 2002.

W. Ke, C. R. Sugimoto, and J. Mostafa. Dynamicity vs. effectiveness: Studying online clustering for scatter/gather. In J. Allan and J. Aslam, editors, *Proc. of the 32nd int. ACM SIGIR conference on Research and development in information retrieval*, pages 19–26. ACM, New York, 2009.

Y. Li, L. Chiticariu, H. Yang, F. R. Reiss, and A. Carreno-fuentes. Wizie: A best practices guided development environment for information extraction. In *Proc. of 50th Ann. Meeting of the Ass. for Comp. Linguistics*, pages 109–114. ACM, 2012.

I. Mierswa. Rapid miner. *KI*, 23(2):62–63, 2009.

D. Shahaf and C. Guestrin. Connecting the dots between news articles. In *Proc. of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–632. ACM, 2010.

G. I. Webb. Self-sufficient itemsets: An approach to screening potentially interesting associations between items. *TKDD*, 4 (1), 2010.

D. Xin, X. Shen, Q. Mei, and J. Han. Discovering interesting patterns through user's interactive feedback. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 773–778. ACM, 2006.