

# Making a Clean Sweep of Cultural Heritage

Antal van den Bosch and Marieke van Erp, *Tilburg University*

Caroline Sporleder, *Saarland University*

**A**s in perhaps no other field of the sciences and humanities, cultural heritage practitioners are profoundly aware of the need to preserve artifacts and knowledge in sustainable ways to guarantee long-term access to heritage material for future generations. Only in the last few years have practitioners

begun to prepare data, information, and knowledge for digital preservation. This is an important revolution: what used to lie behind closed doors, accessible only to curators, can now be shared with the world without harm to the physical objects. Another advantage of digitization is that information about the artifacts—that is, metadata—can be updated, checked, and maintained more easily than before.

To ensure that the digital representations of cultural heritage artifacts are trustworthy, the digitization process must result in high-quality, high-fidelity representations. However, researchers of cultural heritage data as well as curators and heritage data managers realize that the highest levels of quality are not always obtained in practice.<sup>1</sup> When digitized data contains errors, researchers first need to estimate the proportion of errors, or better still, resolve them, which is typically a process that can prove at least as costly as the digitization process itself; otherwise their research is based on flawed data. To a cura-

tor, an artifact can become useless for many purposes if the artifact's provenance—that is, where the artifact comes from and who created it—is unknown. In general, a person searching the data with the goal of collecting complete and correct information typically can't gauge the extent of data corruption. *Precision errors* (returned results that should not have been returned) may be obvious after inspection, but *recall errors* (results that should have been returned but are not) typically remain hidden.

For example, consider a research project studying changes in the geographical distribution of cylinder snakes in Sri Lanka. To obtain an idea of where and when specimens of this family of snakes have been collected in Sri Lanka, the user may query a database for the keywords "Sri Lanka" and the zoological family name. Now if some relevant database entries contain misspellings—for example, "Sri Lanca"—the search won't retrieve them. While a subset of simple typos can be dealt with by fuzzy string matching,

*Automatic error detection is a high priority for cultural heritage data managers and researchers. The authors describe a general approach to cleaning cultural heritage databases.*

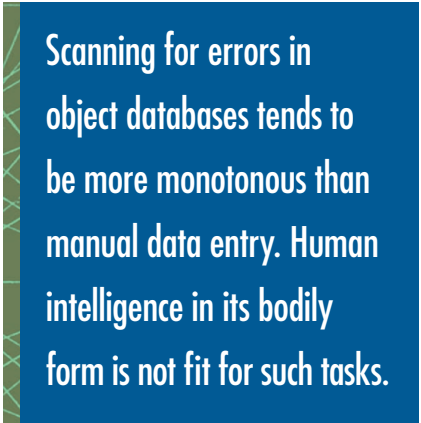
this is not possible for more serious errors—for instance, if the alternative (historical) name “Ceylon” is used for “Sri Lanka” or if the country information is wholly incorrect (for example, “India”). Sometimes the ambiguity proves more subtle; consider the discovery that the painting *Head of Man* at the National Gallery of Victoria, Melbourne, Australia, which art historians previously attributed to Van Gogh, turned out to be by one of his contemporaries. As this was only discovered in 2007, many resources still list the painting as created by Van Gogh. Digital resources are easily adapted to corrections of this type, contrary to printed documents.

Fortunately, an increasing body of research concerns itself with the automatic cleanup of large databases.<sup>2-4</sup> We describe a generic approach that utilizes robust findings from this body of research, and that is applicable to object databases typically found in cultural heritage fields. We focus on the role of automatic database cleanup technology as a strong filter, making the data-cleaning task feasible for the human expert (researcher, curator) by highlighting potential errors and inconsistencies, which the expert can then manually check and correct. A certain level of automation is crucial for the cleanup task. In the case studies we present, the owners of the database express an inability to manually check “everything;” indeed, scanning for errors in object databases tends to be more monotonous than manual data entry. Human intelligence in its limited bodily form is simply not fit for such tasks. Besides that, the manpower to undertake such a project is typically not available, as only few people tend to have enough knowledge about the data in question to perform this task—and these domain experts normally have limited available time. Our goal therefore is to limit the amount of manual correction work in producing trustworthy databases:

our system can drastically reduce the amount of potential errors that an expert should have to inspect.

### Errors in Cultural Heritage Databases

Jonathan Maletic and Andrian Marcus estimate that about 5 percent or more of the information present in manually created databases is erroneous.<sup>5</sup> Many possible causes for errors exist. First, some errors are due to interpretation discrepancies: different people who en-



Scanning for errors in object databases tends to be more monotonous than manual data entry. Human intelligence in its bodily form is not fit for such tasks.

ter data in a single database may have different interpretations of what type of information to enter in particular cells. This tends to hold true for many cultural heritage databases, where the database structure is typically created by the curators or researchers themselves, rather than by professional data managers. Consequently, such databases are often subject to limited quality control: that is, there are no strict (or enforced) guidelines of what information should go in different database cells or how the information should be represented or formatted. Even when the intended database structure is adhered to, database records may be corrupted by typos and copy-and-paste errors, or through optical character recognition errors if the digitization process of the source text was automatic. Also, when a database has

evolved over time, the naming conventions may have changed, as often happens in zoological taxonomies, rendering some information outdated.

To assess the amount and type of errors found in cultural heritage databases, we carried out a study in which a random sample of a database was manually checked for errors. The database comes from the Dutch National Museum of Natural History, *Naturalis*, and contains information about reptile and amphibian specimens in the museum’s collection (see the “Databases” section). Each database record refers to one or more collected specimens; the database fields (columns) encode information about their taxonomic and preservation status, details about the collection, and so on. We identified three types of errors: *content errors*, which are genuinely incorrect values; *spelling errors*, which are orthographically corrupted forms of otherwise correct values; and *wrong-column errors*, which are database values that pertain to the right object in the database and provide correct information but do not fit the database column in which they occur. For the latter two types of errors, which are relatively easy to correct manually, 10,000 nonnumeric database cells were checked by hand (3 percent of the database). This manual inspection took approximately three hours for one person. We detected 123 spelling errors (1.23 percent), and 437 wrong-column errors (4.37 percent), totaling over 5 percent of the errors for these two types alone.

To obtain estimates for content errors, we selected all fields containing taxonomic or geographical information for 257 records (10 columns × 257 records, totaling 2,570 cells). We restricted ourselves to taxonomic and geographical fields because this information can be verified against published resources. Manually checking these cells took nine person hours, yielding 894 content errors (34.8

percent). Of the total number of content errors, 145 were caused by the use of a nonstandard synonym. Many of these errors were repetitive and systematic, and a large portion can also be blamed on an inefficient database structure. The Order column, for instance, often contains the value *Sauria*, which used to be a suborder of the correct value *Squamata* (the use of the term *Sauria* is now deprecated). In the absence of a Suborder column in the database, suborder information is sometimes inconsistently entered in the Order field. While this type of inconsistency will not cause many difficulties for the curators who entered the data (as they are familiar with the ambiguous use of the Order column), it may cause problems for external researchers who want to query the database to find information for their research projects.

This manual error study illustrates how a typical cultural heritage database may contain errors and harmful inconsistencies of various types. The semiautomatic cleanup technique we describe in the next section can detect all three types of error: spelling errors, content errors, and wrong-column errors. Moreover, it deals with all error types in a uniform way, making it unnecessary to run several separate tools over the database—for example, a spelling checker and a content error detector.

### Data Cleaning Using Machine Learning

Trivially but crucially, if 5 percent of the data in the average manually entered database is incorrect, then 95 percent is correct. Hence, statistically speaking, it is possible to cast the data cleaning problem as an outlier detection task. Consider a flat, nonrelational database describing  $N$  cultural heritage objects using  $M$  columns. Each of the  $N \times M$  database cells can be tested for an outlier value. To determine whether a particular value is

an outlier, we exploit the frequent interdependencies between different database columns. For example, the style of an artifact (for example, “black-figure pottery”) may say something about its likely origin (“Greek”). Therefore it is often possible to predict the value of a database cell on the basis of the values of the other cells in that database row. Outliers are cases in which the cell value deviates from the predicted value. The data mining community has adopted this idea as a standard

All disagreements between the classifier’s prediction and the actual value of the database cell are flagged as possible errors.

approach to data cleaning; for example, see the work by Xingquan Zhu, Xindong Wu, and Ying Yang<sup>3</sup> and Jason D. Van Hulse, Taghi Khoshgoftaar, and Haiying Huang,<sup>4</sup> where the typical baseline approach is the distance-based detection of outliers using the classic  $k$ -nearest-neighbor ( $k$ -NN) classifier.<sup>6</sup>

To test whether a given database cell contains an outlier, we train a  $k$ -NN classifier on all other entries (that is, all other rows of the database), where the class that the classifier will predict is the value of the target column, and the input features are all other columns.<sup>7</sup> For instance, if we want to predict the value of the Country field for a particular object  $O$  in a natural history database, we train a classifier on all other objects besides  $O$ , and predict  $O$ ’s value in the Country field on the

basis of a nearest-neighbor search of the vector of values of all other fields of  $O$  against the same vector of all other objects. To check the whole database for errors, we apply the classifier to each cell in turn. All disagreements between the classifier’s prediction and the actual value of the database cell are considered outliers; these are then flagged as possible errors. It is up to the human expert to judge these cases, and determine whether in each case the classifier is correct (so the cell value is incorrect), whether the database is correct, or whether they are in fact both incorrect.

Obviously, many machine-learning algorithms beyond the simple  $k$ -NN classifier are available for this task, but we still opt for using the  $k$ -NN classifier as it is very efficient for incremental and decremental learning, which is needed when holding out a single database entry for classification, and it is also insensitive to the number of classes in the database. If applied to outlier detection in database columns that could contain any number of unique values, it must be able to deal with thousands of classes or more.

When classifying, the  $k$ -NN classifier compares a held-out instance to all the instances stored in memory. The class of the new instance (that is, the database cell currently in focus) is assigned on the basis of the classes of the  $k$  most similar instances, according to a measure of similarity. In our experiments, we used TiMBL (<http://ilk.uvt.nl/timbl>), an efficient implementation of  $k$ -NN.<sup>8</sup> This implementation uses feature weighting to express differences in the predictive power of features in the distance function employed by the  $k$ -NN classifier, and is capable of handling symbolic and numeric features.

### Timpute

Building on TiMBL, we developed a specific data-cleaning system, dubbed Timpute (<http://ilk.uvt.nl/timpute>),

that we can apply to any flat ( $N \times M$ ) nonrelational database suspected of containing errors. As the data-cleaning approach only works for columns that can to some degree be predicted from other columns, Timpute starts with an initial feasibility check of all columns. In this first phase, Timpute randomly splits up the data in a 90 percent training and 10 percent test set, and measures accuracy over a single run. We automatically divide the database columns to be cleaned into two subsets:

- Unpredictable columns, which Timpute will not test for outliers on the basis of any of these criteria: the column only contains a single value (which renders it void of information; its entropy equals 0); the column contains  $N$  values, that is, as many values as there are items in the database, for example unique keys; Timpute can only predict the column values with an accuracy of less than a threshold  $\theta$  ( $0 \leq \theta \leq 1$ ). (To determine the prediction accuracy for a column, Timpute runs a leave-one-out experiment in which each column value is predicted in turn from the other values. The predicted values are then compared to the original values. The accuracy is defined as the proportion of correctly predicted values.)
- Predictable columns, which Timpute will test for outliers if it can predict the column values with an accuracy above  $\theta$ .

Setting a high  $\theta$  ensures that error checking is only applied to columns whose value Timpute can reasonably predict. Indirectly, this also reduces the workload of the expert who is checking the flagged errors, as the cases in which the classifier's predictions deviate from the database are likely to be real outliers. Columns for which the classifier accuracies are low, on the other hand, are probably not suited for

this type of automatic error detection; the sheer amount of flagged errors defeats Timpute's purpose. Practitioners will need to manually clean columns thus discarded or keep them the same.

A partially intended but not guaranteed side-effect of choosing a high accuracy threshold for  $\theta$  is that *object-extrinsic* columns tend to turn up below  $\theta$  (and thus are excluded from prediction), while *object-intrinsic* columns are included in the *predictable columns* subset. Object-extrinsic col-

**Object-intrinsic features are more predictable than object-extrinsic features because the latter category is only accidentally related to the former.**

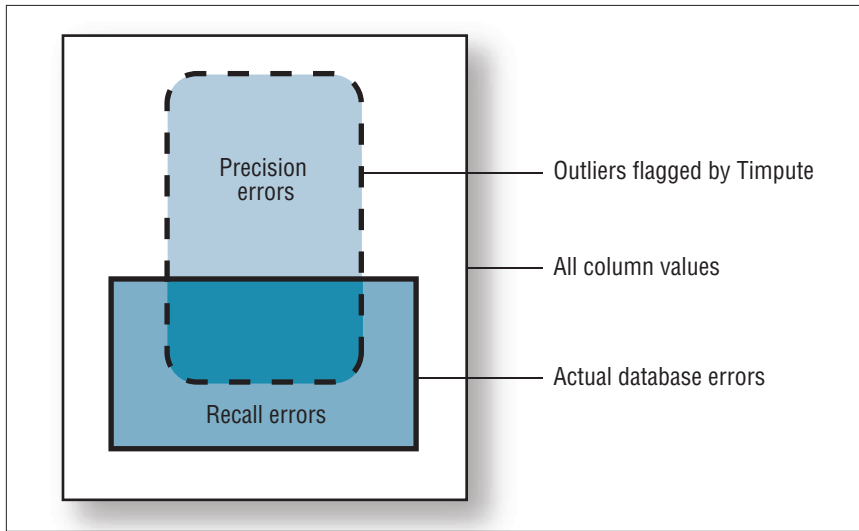
umns contain information only accidentally related to the object at hand, such as a digital video format to which an item of historic film footage has been converted at some point. The digital format bears little or no relation to any of the film footage's intrinsic features; it was chosen at a different point in time, on grounds of often completely unrelated factors. Object-intrinsic features, in contrast, do belong to the object and were usually there when the object was created, discovered, or added to a collection. The example piece of film footage has a director, a voice-over, and a production company, was recorded on a certain type of film, and so on.

Hypothetically, object-intrinsic features are more predictable than object-extrinsic features because the latter category is only accidentally related to

the former. Object-intrinsic features, on the other hand, may be mutually predictive as objects tend to come in families: a television series, a group of excavated farms, a series of paintings created in the same location and time span by the same artist, or a set of animals collected by the same biologist within a single expedition. All of these families share certain intrinsic features that make them partly share the same column values, allowing for the interpolation of one value from the other values of that feature of its nearest neighbors from the same family.

In the second phase after the feasibility check, the actual data imputation by  $k$ -NN classification, as sketched earlier, takes place. Timpute runs leave-one-out experiments for all columns selected in the predictable-columns subset, using  $k$ -NN classification with  $k = 1$ . In each experiment, one of the selected columns is designated as containing the values to be predicted, and all other columns (including the badly predictable features with more than one value and less than  $N$  values) are used as features. Timpute checks all database cells one by one (that is, by leaving out each database entry one by one) and compares each prediction with the actual database cell value. Upon finding a disagreement, Timpute flags the database cell as a potential outlier. Timpute's prediction is added to the flag as a potential correction of the error. A suggestion can be accompanied by a confidence score of the classifier of the proposed value for the user to review. This confidence score could be the distance of the nearest neighbor or the global leave-one-out accuracy measured on the column.

The cultural heritage expert can then decide which of the predictable columns he or she wants to check for errors. This depends on how much time the expert has available at that moment and on how many potential errors are flagged by Timpute for a



**Figure 1. Schematic visualization of precision and recall errors when Timpute flags errors in a database column. The overlapping region (dark grey) represents the portion of correctly identified errors. Some actual database errors are not detected (middle grey), while some flagged outliers are not actual database errors (light grey).**

given column. In addition to  $\theta$ , the expert can set a cutoff for the absolute number of errors to be checked; the system only displays those columns for selection for which the number of detected outliers is below this cutoff. In our experience, and depending on the difficulty of verifying a particular column’s value, a cutoff of around 100 errors is quite manageable for one session. We leave this choice to the user.

**Measuring System Performance: Precision vs. Recall Errors**

An important constraint on our system is the focus on the detection of errors in filled database cells. This implies that we do not attempt to guess the contents of empty cells, and we do not predict that a filled cell should be empty. Handling empty cells is an in-

trinsically difficult problem owing to the fact that a cell can be empty for two extremely different reasons: either it may be empty on purpose, or it may be erroneously empty, that is, it should’ve been filled. For example, a database may contain a column for “special remarks” about an object. Such a column is frequently left empty intentionally. Rather than trying to tackle this intricate issue here, we focus on detecting errors in filled cells only.

In general, the success of error detection and correction is typically measured in terms of precision and recall.<sup>9</sup> Figure 1 illustrates the difference between the two quantities for our task. A database column contains a certain quantity of errors unknown to us, indicated by the dark rectangle with the

continuous border. Next, Timpute flags an overlapping quantity of cells in the column as outliers, indicated by the rectangle with the dashed border. Some of these flagged outliers are also in the set of actual database errors; the other part of the flagged outliers are actually correct values that Timpute should not have flagged. The latter subset constitutes the precision errors made by Timpute on the column. As with search engines, precision can be measured by manually inspecting all flagged outliers; we invite our experts to do this.

On the other hand, Timpute does not detect a portion of actual column errors; these constitute the recall errors. Contrary to the precision errors, recall errors are hard, if not impossible to measure; detecting recall errors implies that the full column is checked for errors, nullifying the reason for using Timpute in the first place. An alternative is to estimate the amount of recall errors by inserting artificial errors, then performing a Timpute sweep and measuring how many of these errors are not flagged.

In general, the amount of recall errors should be minimized, or alternatively put, the recall should be maximal: no real errors should be missed. A large number of precision errors is unwanted in principle as well, as having to check a vast majority of precision errors may strain the expert. A low recall, however, may lead to a situation in which only a very small proportion of the actual errors in the database are found, rendering the whole

**Table 1. Statistics of the four databases used in the study ( $\theta = 90\%$ ).**

Database	Number of items	Number of columns				
		Total	1 value	N values	< $\theta$	> $\theta$
Reptiles and amphibians	16,870	39	0	1	21	17
Plant dialect names	216,903	17	1	0	11	5
Sound and vision	16,138	31	0	1	25	5
Archaeological findings	517	31	1	1	22	7

error detection process futile. Therefore, we aim to maximize recall rather than precision.

### Databases

We tested Timpute on four databases from Dutch cultural heritage institutions. Statistics on the number of items (rows) and columns are listed in Table 1. We describe the databases in more detail in the rest of this section.

### Reptiles and Amphibians

Naturalis maintains various databases cataloging the animal and plant specimens in its collection. One of these contains information on the collection of reptile and amphibian specimens mainly from the Amazon rainforest and Indonesia. It is a flat database in which the animal's place in the zoological taxonomy is recorded, as well as information on where and when it was found, under which circumstances and by whom, and how it is preserved in the museum.

The database contains 16,870 records and 39 columns. Some columns contain simple values such as a single term or a numeric value; others contain longer stretches of free text. Researchers at the museum manually inserted the information which stems from various handwritten and printed sources such as field logbooks, museum registers, labels attached to objects, and publications. Table 2 shows some sample values of one item in the Naturalis database.

### Plant Dialect Names

The Meertens Institute, the Dutch national institute for research and documentation of Dutch language and culture, maintains a database of plant names and their Dutch dialect names. The information comes from field studies dating back to 1885, conducted by the Meertens Institute and other institutions in the Netherlands and Belgium, as well as from books, manuscripts, and dialect dictionaries

**Table 2. Sample columns of one item in the Reptiles and Amphibians database.**

Column	Name value
Special remarks	Died in captivity 23 September 1994
Country	Indonesia
Collector	M.S. Hoogmoed
Species	Pseudolemniscatus
Collection date	06-11-1985
Altitude	1700 m
Author	Roux, 1911
Order	Sauria
Registration number	5529

dating back to 1836.

The database contains 216,903 records, and 17 columns. Each entry refers to one dialect name of a plant, where the columns give additional information, such as the region for which the name has been attested, when it was mentioned for the first time, the Latin and standard Dutch names of the plant it refers to, and two types of keywords to facilitate database searching. It also contains region codes that are linked to maps at the Meertens Institute, so researchers can visualize the usage of a word in its geographical context. The database contains numeric and textual values, rarely exceeding more than two tokens per field.

### Sound and Vision

The Netherlands Institute for Sound and Vision maintains databases holding information about the audio and video recordings in the institute's collection. Information contained in the databases pertains to the contents of a video or radio program, people involved in its creation, and administrative attributes such as when it was created, where and how it is stored in the archive, and who holds the copyright.

The database we worked with contains 16,138 items that each describe one recording, using 31 columns. The values of the fields are mostly textual, sometimes relatively short and fixed, but often containing longer stretches of free text such as summaries of the contents of the object.

### Archaeological Findings

The Dutch National Service for Archaeology, Cultural Landscape, and Built Heritage provides a database describing medieval ceramic objects and their history. It bears similarities to the Reptiles and Amphibians database in that it contains details about how the object was collected and where it can be found in the archive, as well as physical characteristics of the object itself.

The database contains a mere 517 entries, described in 31 columns. Most values are either dates, numbers, or one- or two-word values. Occasionally there are longer strings of text—for instance, in the columns describing physical characteristics of the objects.

### Experiments and Results

In our experiments with Timpute on the four databases, we set the accuracy threshold  $\theta$  to 90 percent, and the error cutoff—that is, the maximum number of potential errors the expert has to check—to 100. If these conditions are met, the column is included in the Timpute process. Next, we show which columns are selected for automatic outlier detection in the four databases, and we provide recall estimates for artificial errors. Then, we present precision measurements based on expert judgments.

### Recall Estimates

Table 3 lists which columns of each database are likely to yield useful

**Table 3. Selected columns of the four databases, their status, number of flagged disagreements, the percentage of unflagged cases, and estimated recall.**

Database	Column	No. values	Intrinsic/Extrinsic	No. flagged	% unflagged	Est. recall
Reptiles and Amphibians	Country	71	I	68	0.96	1.00
	Inventory no.	3	E	2	1.00	0.99
	Printed	6	E	67	1.00	1.00
	Publication	86	I	57	0.98	0.59
	Recorder	8	E	87	0.99	1.00
	Reference no.	2	E	1	1.00	0.00
	Subspecies	277	I	97	0.96	0.43
	Class	3	I	10	1.00	0.99
	Order	13	I	88	0.99	1.00
Plant Dialect Names	Latin name	4	I	1	1.00	0.88
Sound and Vision	Collection	2	I	3	1.00	0.97
	Copyrights	4	I	6	0.98	1.00
	Maker function	29	I	36	0.99	1.00
	Source	3	I	1	0.99	0.60
Archaeological Findings	Base technique	22	I	36	0.93	1.00
	Dating	17	I	14	0.97	0.50
	Literature	21	E	42	0.92	0.48
	Part	16	I	2	1.00	0.96
	Structural hardness	4	I	12	0.98	1.00
	Structural porosity	5	I	29	0.94	1.00
	Year	12	I	1	1.00	1.00

results. In this table we have also incorporated another selection criterion for manually checking the data—namely, whether the column is object-intrinsic (I) or object-extrinsic (E). The table shows that Timpute mainly selects intrinsic columns for semiautomatic error correction.

Timpute's initial feasibility check uncovers that the plant dialect names database has very few intrinsic and predictable columns. The only way to check this data is by accessing the original sources.

As discussed earlier, the recall rate of an error detector is hard, if not impossible, to determine. As an alternative, we estimated recall by introducing errors artificially and determining what percentage of these artificial errors was detected. For each field in the target columns, we changed the values

of 5 percent of the entries. In these entries, the original value was replaced by a randomly selected value occurring in the same column elsewhere in the database. The new value was selected with a uniform probability for all values. Of course, this method can only provide an estimate of the true recall, as it is possible that real errors are distributed differently; that is, some values may be more easily confused by humans than others. With this caveat in mind, it is clear that Timpute is generally quite good at spotting errors, with recall rates for many columns well above 95 percent. Essentially this means that after running Timpute and having the flagged cell values checked by an expert, the vast majority of errors in these columns have indeed been found and corrected, resulting in a much cleaner database.

### Precision Analysis

In Table 4, the number of flagged cases (for example, disagreements) between Timpute and the reptiles and amphibians database is shown on three of its object-intrinsic columns above the  $\theta$  threshold and error cutoff. Table 4 also shows that the flags raised by Timpute are genuine errors in roughly half of the cases—that is, 46 out of the 88 flagged cases for the Order column, or a precision of 0.52. In other words, roughly half of Timpute's flags are false hits; the human annotator will find a genuine error in about half of the flagged cases. This would seem a reasonable ratio of hits and misses.

Overall, our results show that Timpute can detect errors in cultural heritage databases with high recall and reasonable precision, enabling users to improve data quality greatly.

**Table 4. Error detection performance, in terms of the number of flagged errors, divided further into the number of correctly detected errors (Detected) and the precision in parentheses, classifier errors (Timpute error), and cases unassessable without access to the object (Indeterminable).**

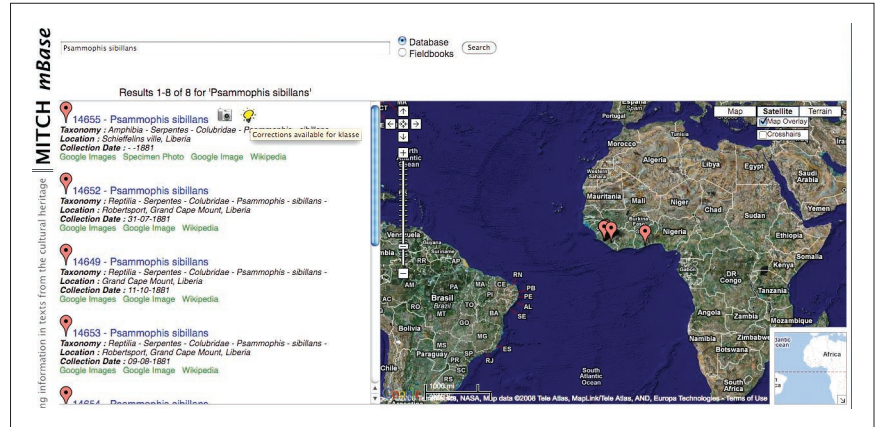
Column	No. flagged	Detected (precision)	Timpute error	Indeterminable
Class	10	6 (0.60)	4	–
Order	88	46 (0.52)	39	3
Country	68	26 (0.38)	37	5

### Interacting with Timpute

We integrated Timpute with a higher-level information system prototype that we developed for Naturalis, called mBase (<http://aether.uvt.nl/mBase>). mBase offers a search-and-browse interface, providing researchers with easy and intuitive access to specimen databases. Timpute is integrated in mBase, which supports linking to various other resources, such as online geocoding of information and photo collections. The interface offers standard keyword search on the whole knowledge base or on specific fields.

Figure 2 gives an example of some results returned by mBase to the query “*Psammophis sibillans*.” For each record the user can see whether it contains any values that are flagged by Timpute as potential errors, indicated by the yellow lightbulb icon. When the user hovers over it, he or she can see for which field Timpute suggests corrections. The camera icon next to the lightbulb indicates that a photo of the specimen in the collection is available. The tool also provides a link to an image of the original register or field log-book, or the transcribed text of these, if available, so that the user can check whether the suspicious value stems from the paper resource or was introduced during conversion to the database. The visualization of the data on a map may also help highlight errors in the data. For example, a specimen of a species different from all other examples collected on a continent may suggest inconsistent information.

When the user clicks on the record, a detailed database-like view of the specimen is presented (see Figure 3). Here the user can review all information available on the specimen in the



**Figure 2. Screenshot of results returned on a keyword query to the Reptiles and Amphibians database at Naturalis by mBase. A lightbulb indicates a suggested correction. Location descriptions are mapped to coordinates, visualized in Google Maps.**

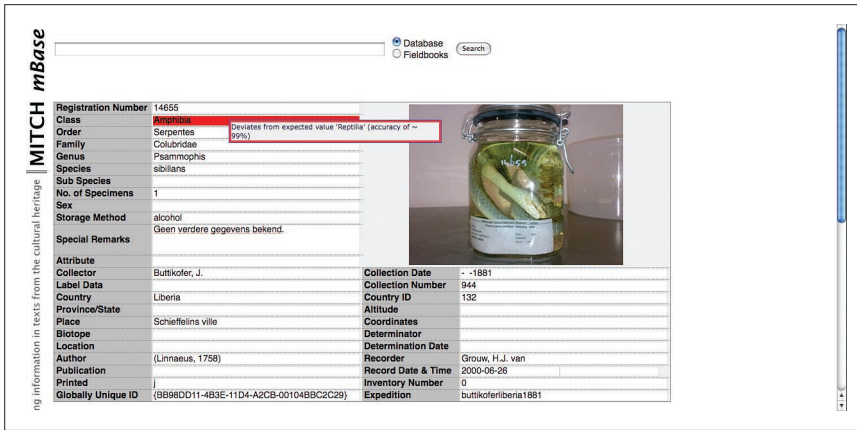
system, and submit changes or additions. Timpute’s output is integrated in this view. When Timpute flags a field, it indicates the cell with a red background (the color hue can be used to express confidence in the error). If the user hovers over the cell, the interface generates a pop-up information window that explains that the cell value deviates from what Timpute expects. In the figure, a snake has been incorrectly classified as belonging to “Amphibia,” whereas it should belong to “Reptilia.” Timpute’s overall percentage of 99 percent unflagged cells on this column (see Table 3) is reported here to provide an indication of Timpute’s confidence.

### User Experiences: Provenance and Confidence

In our discussions with domain experts who worked with Timpute, we found that most feedback relates to Timpute’s precision errors. Recall errors remain invisible, and thus unper-

ceived. Furthermore, the time savings with Timpute for automatic error detection as compared to manual error checking of the whole column was not always apparent. This is similar to searching the Web with search engines: the user’s perception is drawn toward what is actually presented by the engine, whereas recall and a potentially vast amount of missed matching pages remain unseen and out of the user’s perception.

With respect to precision errors, experts tend to remark that many correctly flagged errors are not severe database errors but rather minor mistakes in consistency, such as the inconsistent spelling of a person’s name. Timpute is able to pinpoint these consistency errors quite well, making it clear to the expert users that they should pay more attention to the consistent use of naming and terminology during manual data entry. A data entry interface that would guide the user by proactively proposing controlled and consistent



**Figure 3. Details of an animal specimen database entry result returned by mBase. A snake has been incorrectly classified as belonging to the taxonomic class “Amphibia.” This error, found by Timpute, is colored, and Timpute’s suggestion and confidence indication are displayed.**

cell values would aid the consistency of databases considerably.

Over time we also gathered skeptical remarks, notably by researchers of cultural heritage data expressing disbelief that a machine could automatically spot errors, as it does not possess domain knowledge. Counterarguments that it does possess at least some knowledge implicitly by its capacity to mine the mutual predictability of columns in the database are usually unconvincing. Therefore, in addition to the semiautomatic presentation of errors to users, we are investigating whether it would help to make the implicit expertise of our system explicit, to counter the skepticism. Our *k*-NN classifier does not have the capacity to explain why it makes certain decisions, but it can show the nearest neighbors it found to come to a prediction—this may help a human expert understand why Timpute suggested a correction.

Another often-heard remark is that domain experts (curators and researchers alike) stress the importance of never deleting any information, even if it is incorrect. Original cell values that are flagged and corrected should always be retained; in addition, flags should be labeled as unchecked or checked by a human expert. Also, if the suggested correction is associated with a confidence, either from the

machine learner or from the expert labeler who inspected the flagged outliers, this should be stored, too. This calls for the application of proper version control on top of the use of database management systems, offering the possibility to always roll back to earlier versions of the database.

The data-cleaning approach presented here offers a functionality that is broadly recognized as much-needed and vital for quality control of digitized cultural heritage data. Our approach is generic, in the sense that practitioners can apply it to any domain that has a substantially large database in need of cleaning (containing, as illustrated, at least several hundred items), but does not require the availability of external digital resources such as normalized lists of object names or classifications. The system reduces the workload on domain experts when correcting errors, by zooming in on the errors, with high estimated recall.

Our basic knowledge-free system can be expanded quite easily by other, more specific, correction methods and by domain-specific knowledge. For instance, for detecting and correcting spelling errors, we intend to add an initial spell-check phase that will use standard resources where available

(such as taxonomies, dictionaries, and domain-specific lists of names).

Currently, the system is limited to handling flat databases; future versions will have the ability to deal with relational databases. We also found that for the prediction of certain columns, Timpute relies heavily on certain other columns. Errors in these columns will tend to cause Timpute to generate erroneous predictions as well; therefore, we will investigate an incremental version, taking corrections to other columns into account. Finally, we aim to develop an online version of Timpute, running in the background while information is added to databases, to notify users immediately of possibly erroneous values as they are entered. We will further develop and test the mBase search and correction prototype at Naturalis, after which we intend to integrate it with the museum’s existing information architecture. Testing will involve contrastive processing-time measurements, to provide estimates of actual time savings offered by Timpute.

In sum, we see a rising awareness in the necessity of digital data quality control in the cultural heritage field. We believe cleaning digital data should constitute a standard first step in information processing at cultural heritage institutions. Furthermore, it is vital to keep the human experts in the loop; it is in their own interest that their data is improved, and our approach allows them to take this hurdle that has so far been a humanly infeasible task. ■

**Acknowledgments**

We thank Tijn Porcelijn and Steve Hunt for their programming assistance. We are grateful to the donors of the four data sets: Guus Lange and Pim Arntzen, who both provided us with expert feedback, Luit Gazendam, Hennie Brugman, Frans Wiering, and Maarten van der Peet. This research was funded by the Netherlands Organization for Scientific Research, under the Continuous Access to Cultural Heritage (Catch) program.

## References

1. A. Chapman, *Principles and Methods of Data Cleaning: Primary Species and Species Occurrence Data*, ver. 1.0, tech. report, Global Biodiversity Information Facility, 2005.
2. J. Kubica and A. Moore, "Probabilistic Noise Identification and Data Cleaning," *3rd IEEE Int'l Conf. Data Mining (ICDM 03)*, IEEE CS Press, 2003, pp. 131–138.
3. X. Zhu, X. Wu, and Y. Yang, "Error Detection and Impact-Sensitive Instance Ranking in Noisy Datasets," *Proc. 19th Nat'l Conf. Artificial Intelligence (AAAI 04)*, AAAI Press, 2004, pp. 378–383.
4. J. Van Hulse, T. Khoshgoftaar, and H. Huang, "The Pairwise Attribute Noise Detection Algorithm," *Knowledge and Information Systems*, vol. 11, no. 2, 2007, pp. 171–190.
5. J. Maletic and A. Marcus, "Data Cleansing: Beyond Integrity Analysis," *Proc. Int'l Conf. Information Quality (ICIQ 00)*, MIT Press, 2000, pp. 200–209.
6. T.M. Cover and P.E. Hart, "Nearest

## THE AUTHORS

**Antal van den Bosch** is a full professor of computational linguistics and artificial intelligence at Tilburg University. His research interests include memory-based natural language processing, machine translation, and proofing tools. Van den Bosch received his PhD in computer science from the University of Maastricht. He's a member of ACL. Contact him at antal.vdnbosch@uvt.nl.

**Marieke van Erp** is a PhD student at Tilburg University. Her research interests include domain specific natural language processing, information extraction and ontology learning. Van Erp received her MS in language and artificial intelligence from Tilburg University. She's a member of IEEE and ACL. Contact her at m.g.j.vanerp@uvt.nl.

**Caroline Sporleder** is a research group leader at Saarland University. Her research interests include computational models of discourse and semantics, text mining, and cross-lingual and cross-domain language processing. Sporleder received her PhD in Informatics from the University of Edinburgh. She's a member of ACL. Contact her at csporleder@coli.uni-sb.de.

Neighbor Pattern Classification," *IEEE Trans. Information Theory*, vol. 13, no. 1, 1967, pp. 21–27.

7. C. Sporleder et al., "Spotting the 'Odd-One-Out': Data-Driven Error Detection and Correction in Textual Databases," *Proc. EACL 2006 Workshop Adaptive Text Extraction and Mining (ATEM 06)*, Assoc. Computational Linguistics,

2006, pp. 40–47.

8. W. Daelemans and A. van den Bosch, *Memory-Based Language Processing*, Cambridge Univ. Press, 2005.
9. M. Reynaert, *Text-Induced Spelling Correction*, PhD dissertation, Computational Linguistics and Artificial Intelligence Research Unit, Tilburg Univ., 2005.

**Call for Articles**

**IEEE Pervasive Computing**

seeks accessible, useful papers on the latest peer-reviewed developments in pervasive, mobile, and ubiquitous computing. Topics include hardware technology, software infrastructure, real-world sensing and interaction, human-computer interaction, and systems considerations, including deployment, scalability, security, and privacy.

**Author guidelines:**  
[www.computer.org/mc/pervasive/author.htm](http://www.computer.org/mc/pervasive/author.htm)

**Further details:**  
[pervasive@computer.org](mailto:pervasive@computer.org)  
[www.computer.org/pervasive](http://www.computer.org/pervasive)

**IEEE pervasive COMPUTING**  
MOBILE AND UBIQUITOUS SYSTEMS