

Cleaning and Enriching Research Data on
Reptiles and Amphibians.
The MITCH Pilot Project and “nulmeting”

Induction of Linguistic Knowledge Research Group
Technical Report ILK 06-01

Caroline Sporleder, Marieke van Erp, Tijn Porcelijn, Antal van den Bosch
ILK/Language and Information Science
Tilburg University, P.O. Box 90153,
5000 LE Tilburg, The Netherlands
{C.Sporleder,M.G.J.vanErp,M.Porcelijn,Antal.vdnBosch}@uvt.nl

Pim Arntzen, Erik van Nieukerken
Naturalis, National Museum of Natural History
Darwinweg 2, 2333 CR Leiden, The Netherlands
{Arntzen,Nieukerken}@naturalis.nl

February 2, 2006

1 Introduction

This document describes a pilot study undertaken as part of the MITCH project. MITCH stands for “Mining for Information in Texts from the Cultural Heritage” and is part of the NWO¹ funded CATCH programme (Continuous Access To Cultural Heritage), whose aim it is to develop new technologies to improve access to cultural heritage.

MITCH is a cooperation between the Language and Information Science Department of Tilburg University and Naturalis, the Dutch National Museum of Natural History in Leiden. The project aims at researching and developing techniques to automatically extract information and knowledge from field books and other textual data that describes Naturalis’s vast collection of specimens. Ultimately these techniques could assist experts in the field in their research; for example, by supporting sophisticated querying of the data, by signalling interesting interdependencies or inconsistencies, or by automatically linking data to relevant background knowledge, such as maps, climate charts or zoological taxonomies.

The purpose of the pilot project, which ran from 1 September to 31 December 2005, was to tackle a subproblem relevant to the MITCH project, to develop and implement a solution, and to test its feasibility on a small dataset. We chose to look at various data clean-up and enrichment techniques and test them on a database which contains information about some of the Amphibian and Reptile specimens in Naturalis’s collection. Such a database is an important resource for researchers in the field, especially if the contents can be systematically searched and queried. Information retrieval results, however, can be adversely affected by errors and inconsistencies in the data. For example, a biologist interested in finding out about the different biotopes in which a given species was found, might query for the content of the BIOTOPE column for all specimens of that species. Whenever information about the biotope was entered in the wrong column, that particular record will not be retrieved by such a query. Similarly, if a record lists the wrong species, it will also not be retrieved. On the other hand, information retrieval can be improved by enriching the data; for example, by identifying and marking all named entities, which would allow for more sophisticated queries, or by (semi-)automatically completing incomplete records. Data clean-up and enrichment are thus important subtasks of information extraction in general and of the MITCH project in particular.

For the pilot project, we looked at three clean-up and enrichment tasks: named-entity tagging, data completion, and error correction. For all three tasks we focused on developing techniques which are data-driven and knowledge-lean. Such techniques work in the absence of background knowledge (e.g., knowledge about the domain or the structure of the database) and instead rely on the data itself to provide the necessary cues about inconsistencies or likely completions for incomplete records. Furthermore, none of the techniques we developed requires manual annotation of training data. Instead, we exploited the semi-structured nature of the database and derived all training examples automatically. The advantage of knowledge-lean methods is that they can be ported to new datasets and domains easily. Hence, we believe that the techniques we developed during the pilot project cannot only be applied to other databases from the natural history domain but also to cultural heritage databases in general.

We aimed for language independence and tried to avoid or reduce the use of language specific processing tools, such as part-of-speech taggers or chunkers. This was motivated by the fact that the database we used in the pilot project is multi-lingual and contains strings of text in various languages. We believe that this may be the case for many cultural heritage databases. If textual data-cleaning and enrichment methods are to be useful for such databases, they should be able to process all text strings,

¹Netherlands Organisation for Scientific Research (<http://www.nwo.nl>).

not only those in the majority language.

This report gives a detailed description of the work we undertook and of our findings. The next section (Section 2) gives an overview of the data collections we plan to use throughout the MITCH project as a whole and describes the database we used for the pilot project in more detail. In Section 3, we discuss the two enrichment tasks we addressed (named entity tagging and data completion). Section 4 focuses on data cleaning; in particular we describe two different error correction methods which we implemented and tested. The concluding section (Section 5) briefly summarises our findings and gives an overview of ongoing work and research planned over the next few months.

2 Data

During the MITCH project, we intend to make use of a variety of textual and semi-textual resources available at Naturalis. The next section gives a brief overview of these resources and their current state of digitisation. In the pilot project, we developed processing tools for a manually created specimens database. This database is described in more detail in Section 2.2.

2.1 Overview of Textual Resources at Naturalis

Field Books and Register Books *Field books* are the logs that biologists keep while they are 'out in the field' gathering specimens for Naturalis's collection. The entries contain information about the circumstances under which a specimen was collected and also include administrative details, such as field book internal registration codes which ensure that the collected items can be linked back to the field book entries. *Register books* link the collected specimens with Naturalis internal register numbers, thus ensuring that specimens can be found once they have entered the collection. They are typically more carefully written than field books. There is some overlap with field books in terms of the information that is provided, though some details present in the field books may be lacking from the register books (e.g., extensive observations about the biotope in which a specimen was found) and vice versa (e.g., the Naturalis-internal register number is lacking from the field books). In both types of books, the order in which the information is provided is usually relatively fixed. For example, the register number comes first, then the taxonomic classification etc. At the moment, the field and register books are only available on paper and as digital photographs and are thus not yet accessible for automatic information extraction methods. However, work is currently underway to manually transcribe a significant subset of these books. This work will be carried out by Combiwerk² over the next few months. Once the first batch of transcribed books becomes available we will be able to use them in our research.

Register Labels These are the (physical) labels that are attached to the specimens and/or the containers in which the specimens are stored. In addition to the register number and the species name they often also provide additional information, for example, about who collected the specimen. Like the field and register books, the register labels are not yet available in transcribed form. However, at least some of them are included in the data set that will be manually transcribed by Combiwerk.

Specimen Databases For some animal groups, specimen databases were created manually from the information contained in the field and register books. Three such databases are currently available to us, containing information about Reptiles and Amphibians, Mites, and Butterflies. The first database

²<http://www.combiwerk.nl/>

(“Reptiles and Amphibians”) was used in the pilot project and is therefore described in more detail in Section 2.2 and in the appendix.

Published Scientific Papers Scientific papers written by Naturalis researchers about specimens in the collection provide another source of data for the MITCH project. Where these papers are available in electronic form (e.g., as pdf files), they can be processed automatically, subject to certain restrictions (e.g., minor conversion problems from pdf to plain text). Relevant papers written by other researchers in the field may also be used as background information.

Photographic Collections For some specimens or groups of specimens, digitised photographs are available. These are normally accompanied by textual information, which provides a link to the corresponding specimens.

2.2 The Reptiles and Amphibians Database

In the pilot project, we used a manually created database containing information about (parts of) Naturalis’s collection of reptile and amphibian specimens. This database was created by biologists for (biological) research purposes and as an electronic archive of parts of the Reptiles and Amphibians collection. It is well-maintained and regularly updated. While the database was mainly created as an archive and for small-scale querying, it is nonetheless a first step towards a fully-fledged information retrieval system in which information is automatically combined from different sources (register labels, field books etc.) and enriched with background information (e.g., links to climate charts, maps, or photographs). Data that is linked in this way is potentially a very valuable resource that will not only support current (biological) research at Naturalis, but will also benefit the work of outside and future researchers. It is the aim of the MITCH project to go some way towards this goal of automatic data linking and enrichment.

Choosing a database for our pilot project was motivated by a number of factors. First, while the ultimate goal of the MITCH project is to develop data mining and information extraction tools for plain text resources, especially field and register books, these resources are not yet available in a format that permits automatic processing (see Section 2.1). Second, the challenges that arise with data mining from databases, such as dealing with noisy data, are similar to those that will have to be addressed when working with field book transcriptions; however, these challenges will be on a much smaller scale. For instance, a database that was carefully put together by biologists will inevitably contain less noise than field book transcriptions created by non-experts. Furthermore, the implicit structure that is contained in the field and register books is made explicit in a database. This made the specimen databases an ideal source of data for the pilot project. We chose the Reptiles and Amphibians database because it is much larger than any of the other two databases available to us. Furthermore, it is also the most interesting one for automatic data cleansing, because, unlike the other databases, it has been worked on by several people and thus contains more inconsistencies and errors.

The Reptiles and Amphibians database contains currently 16,870 entries and 47 columns. Each entry represents a stored object (which may contain more than one animal, e.g., jars with more than a single specimen). The fields provide a variety of information about each item, such as the name of the person who identified the animal, the data available on the label of the object, its inventory number, species, subspecies, sex, where and when the animal was found, etc. Many fields are textual; some text fields only contain single words or expressions of a certain type (e.g. proper names), while others contain longer stretches of text, such as “special remarks” or descriptions of the biotopes where the animals were caught. The database is multi-lingual, with Dutch being the most frequent

language by far, followed by English; Portuguese is also occasionally used, as well as Latin for animal names. In principle, there is no limit to which languages can occur in the database. For example, the PUBLICATION column often contains texts in languages other than Dutch or English.

The database is relatively sparse: ten of the 47 columns are unused and only just under 40% of the cells are filled. There is a relatively large variance in the number of different values in each column, ranging from three for CLASS (i.e., *Reptilia*, *Amphibia*, and a remark pointing to a taxonomic inconsistency in the entry) to over 2,000 for SPECIAL REMARKS.³ On the other hand there is also some repetition of cell contents, even for the free text columns, which often contain formulaic expressions. For example, the strings *no further data available* or *(found) dead on road* occur repeatedly in the special remarks field. The columns in the database and their meaning are listed in the appendix.

3 Data Enrichment

Data enrichment subsumes tasks such as named entity tagging, automatic link creation (e.g., identifying whether two names refer to the same person), or automatic completion of incomplete records. We addressed two enrichment tasks: named entity tagging and automatic data completion (i.e., filling empty fields in the database). The techniques we developed for these tasks are described in sections 3.1 and 3.2, respectively.

3.1 Named Entity Tagging

Named entity tagging refers to the task of identifying named entities in text and assigning them to the appropriate entity category, e.g. PERSON or GEOGRAPHICAL LOCATION. Named entity tagging is an important subtask for information extraction and retrieval. For example, one might want to query a data set for a list of all locations at which specimens of a particular species were found. If the data is represented as a database, the database structure already provides implicit information about named entities. For example, text strings in a column entitled LAND (country) should be of type LOCATION. However, many textual databases also contain several “free text” fields, such as BIJZONDERHEDEN (special remarks), which would benefit from named entity tagging.

Named entity tagging is a well studied area, with many implemented systems obtaining average F-scores well above 80% (e.g., Florian et al. [2003]). However, generic named entity taggers are typically trained and tested on texts from the news domain. Porting them to cultural heritage data may not always be an optimal solution, not only because there may be domain-specific differences in language use, but also because new entity classes may be needed for the cultural heritage domain (cf. Bontcheva et al. [2002]). For example, for archaeological data it would be useful to have categories like ARTEFACT or SITE. If a generic tagger is applied to textual *databases*, the problem is likely to worsen, due to the fact that text in database fields may be very different from “normal”, unstructured text; for example, text fields in databases often do not contain complete sentences.

Given that porting a generic tagger to our database may not be a good solution, we investigated whether it is possible to bootstrap a domain-specific named entity tagger by exploiting the fact that some information about named entities is already implicit in entity specific database fields. This makes it possible to automatically build gazetteer lists for different named entity types, which can then be used to create training examples for a domain-specific named entity tagger for our database. Similar bootstrapping approaches for named entity recognition have been proposed before, though to

³Note that this field is only filled for a minority of the entries.

our knowledge not for textual databases. We review other work in this area in the next section. Our approach is described in more detail in Section 3.1.2.

3.1.1 Related Work

Over the past few years there has been a lot of research into named entity recognition and other semantic labelling approaches that do not require a large set of manually annotated training examples. A number of approaches exploit the redundancy between word internal and contextual cues (Collins and Singer [1999], Cucerzan and Yarowsky [1999]). An example of a word internal cue is the fact that a capitalised word starting with the letters “Mc” is likely to belong to the PERSON class, as in *McLeod*. An example of a contextual cue is that a word which is followed by *Ltd.* is likely to be of type ORGANISATION. These two types of cues can be combined in a co-training loop. The basic algorithm starts with a small seed list which is used to label examples in an unannotated corpus. These then form the basis for extracting contextual classification rules which are again applied to the corpus to label additional examples from which word internal rules can be generated and so on. Phillips and Riloff [2002] also proposed a co-training approach but instead of relying on word internal and contextual cues they exploit the fact that common and proper nouns of the same semantic class tend to co-occur in certain syntactic constructions, such as appositives. For example, if the expression *John Seng, the financial analyst* is encountered and it is known that *John Seng* belongs to the PERSON class then it can be inferred that *financial analyst* also belongs to that class. This in turn allows one to infer that *Peter Smith* in *Peter Smith, the financial analyst* is also of type PERSON. Lin et al. [2003] present a bootstrapping approach in which multiple semantic classes are learnt simultaneously on the basis of positive and negative examples. Finally, Buchholz and van den Bosch [2000] automatically generate large seed lists from the internet and use these to label training examples in a corpus. A classifier is then trained on the training data in a single training step. The approach we present in the next section is similar. However, instead of generating seed lists from the web, we exploit the database structure to create gazetteer lists.

3.1.2 Bootstrapping a Named Entity Tagger

We used three different named entity classes: location (LOC), person (PER), and taxonomic (TAX, e.g. species names). The gazetteers for the three entity classes were obtained from 14 entity specific fields, e.g. VERZAMELAAR (collector) for PER, PLAATS (finding place) for LOC, and FAMILIE (family) for TAX. Table 1 shows the fields from which we extracted, together with the entity type and an example entry for each field. Direct extraction from these fields leads to some noise; for example, person fields often contain more than one entity or additional material. For instance the AUTEUR (author) field, which provides information about the author of the first comprehensive description of a species, may contain the string “Cole & Dessauer, 1993”, where “1993” refers to the year in which the description was published. We implemented a few simple rules to split multiple expressions and remove additional material. We also used a small list of stop words to identify and remove expressions that are not of the required type. For instance, the DONATOR field sometimes contains entities that are organisations (e.g. *Museum of Vertebrate Zoology*); we identified these by looking for words like *museum* or *collection*. Finally, we generalised person names, e.g., if *M.S. Hoogmoed* was found, we also added *Hoogmoed*, *M.S.* and *Hoogmoed* to the list. Overall we extracted 1,009 expressions of type PER, 3,568 expressions of type LOC and 2,167 expressions of type TAX.

While the gazetteers can be used to identify entities directly via look-up, they can also be used to automatically extract and label training examples (i.e., named entities in context) for a named entity

Field	Example	Type
PLAATS	Bigisanti beach	LOC
PROVINCIE/STAAT	Marowijne	LOC
LAND	Indonesia	LOC
AUTEUR	Daudin, 1802	PER
VERZAMELAAR	Hoogmoed, M.S.	PER
DONATOR	P.J.M. Maas	PER
DETERMINATOR	M.S. Hoogmoed	PER
RECORDER	Grouw, H.J. van	PER
SUB-SPECIES	marmoratus	TAX
SPECIES	pseudolemniscatus	TAX
GENUS	Anolis	TAX
FAMILIE	Polychrotidae	TAX
ORDE	Sauria	TAX
KLASS	Reptilia	TAX

Table 1: Fields from which gazetteers were extracted

	PER	LOC	TAX	OTHER	overall
test set	90	42	68	2,590	2,790
training set	555	265	253	22,972	23,545

Table 2: Distribution of token labels for test and training sets

tagger (see, the overview of related work in the previous section). Once such a tagger has been trained, it should be able to exploit contextual and word internal cues to identify named entities even when they are not in the original gazetteer lists.

We used one of the free text fields (BIJZONDERHEDEN, special remarks) to obtain training and test data for the experiments. When we carried out the experiments, the BIJZONDERHEDEN field contained around 2200 filled cells. We randomly selected 10% of these for testing and the remaining 90% for training. The text in the test and training sets was then tokenised⁴ and the test set was manually annotated with named entity information. In the training set, all expressions for which an exact match was found in one of the gazetteer lists were automatically labelled with the corresponding tag; all other tokens were labelled as OTHER.⁵ Table 2 shows the number of tokens in the data sets and the number of tokens for each entity type and for non-entities (OTHER).

The tagged examples were then used to train a memory-based classifier (TiMBL, Daelemans et al. [2004]) to identify named entities in free text fields. We employed 66 features, encoding the following information for the target token, the two tokens to the left and the two tokens to the right:

⁴We used a rule-based tokeniser for Dutch developed by Sabine Buchholz.

⁵This labelling strategy potentially results in some false negatives, i.e. expressions which belong to one of the three entity classes but are not found in the gazetteers and thus labelled as OTHER. We also experimented with a more conservative labelling strategy in which we only labelled expressions which were found in the gazetteers, together with two tokens to the left and right (which were labelled as OTHER if they were not found in the gazetteer). As this training set does not contain many items of type OTHER, we boosted it by adding the content of the BIOTOOP field, which is unlikely to contain any of our entity types and thus can be relatively safely tagged as OTHER. However, we found no significant difference in performance between the two labelling strategies.

- the token (t) itself
- whether t is capitalised
- whether t contains one or more digits
- whether t contains one or more letters
- the length of t in characters
- whether t contains punctuation
- whether t contains a hyphen
- whether t is utterance/sentence-initial
- whether t is utterance/sentence-final
- the first 4 letters of t
- the last 4 letters of t
- whether t looks like an initial (e.g. E., EM etc.)
- whether t occurred in one of the gazetteers

TiMBL’s parameters (similarity metric, number of nearest neighbours etc.) were set by running a heuristic search algorithm on the training set (wrapped progressive sampling, see van den Bosch [2004]).

We then applied the trained classifier to the test set. For comparison, we also tagged the test set with two baseline taggers: a simple look-up tagger using our automatically created gazetteers and a generic named entity tagger for Dutch [Bogers, 2004]. The generic tagger was trained on the CoNLL 2002 shared task data [Tjong Kim Sang, 2002] and makes use of a variety of features, such as orthographic information, morphological information, part-of-speech tags. The tagger distinguishes five classes (PERSON, LOCATION, ORGANISATION, MISCELLANEOUS NAMES, and OTHER). It is reported to perform at around 70% average F-score on the CoNLL 2002 shared task data.

We evaluated all three taggers on a token-by-token basis, calculating precision (P), recall (R) and F-score (F).⁶ As the generic tagger does not have the entity category TAXONOMIC, we only evaluated this tagger on PERSON and LOCATION. Table 3 shows the results.

	Look-up			Database Trained			Generic		
	P %	R %	F %	P %	R %	F %	P %	R %	F %
PER	84.42	72.22	77.84	79.01	71.11	74.85	34.12	32.22	33.14
LOC	65.51	45.23	53.52	56.00	33.33	41.79	75.00	21.43	33.33
TAX	97.56	58.82	73.39	86.11	45.59	59.62	—	—	—
OTHER	97.47	99.46	98.45	97.28	99.46	98.35	—	—	—

Table 3: Results for the three taggers

The results show that the simple look-up tagger performs surprisingly well, with an average (macro) F-score of 75.80%. This is in line with previous research which found that good gazetteer lists combined with a simple look-up strategy can be remarkably successful [Maynard et al., 2004,

⁶*Precision* is the number of correctly tagged tokens for a given entity category divided by all tokens which were tagged with this category. *Recall* is the number of correctly tagged tokens for a given entity category divided by the number of all tokens with this category in the manually annotated data. The *F-Score* is defined as $\frac{2PR}{P+R}$.

Mikheev et al., 1999, Palmer and Day, 1997]. The relatively high recall values for the entity classes (and in particular for PER) suggest that there is a lot of overlap between the entities mentioned in the entity specific database fields and those in the free text fields. In other words, many entities that are mentioned in the free text fields are also mentioned in the entity-specific fields. The precision is also very high, as one would expect for a look-up strategy. There are two reasons why precision is not perfect. First some tokens are ambiguous between two entity classes. For example, the token *Virginia* is not only a person name but also a location and part of a taxonomic name (*Virginia striatula*). Similarly, *Iguana* usually falls in the category TAX, but in the expression *Reptilien Zoo Iguana* it should be tagged as OTHER. As the look-up tagger does not use context, it cannot disambiguate these cases. Another reason for the imperfect precision is that our automatically built gazetteer lists are not entirely noise-free. For example, the gazetteer list for LOC contains the word *vindplaats* (*finding place*), due to the fact that one of the location columns contained the expression *vindplaats: Garoet, Java* and this was not filtered out during the gazetteer construction.

Compared to the look-up tagger, the database trained classifier performs less well. It achieves an average (macro) F-score of 68.65%, and for all three entity classes both recall and precision are lower than for the look-up tagger. This can be due to several factors. First, it is possible that our training set was simply too small, especially with respect to the three entity classes (with 555 tokens of type PERSON, 265 of type LOCATION and 253 of type TAXONOMIC, see Table 2). It is also possible that differences between the test and training data had a negative effect. For instance, person names in the entity specific fields typically do not contain initials, hence initials are not learnt as being part of a name and are thus frequently misclassified as OTHER in the test set. We experimented with various strategies to improve the results obtained by the database trained tagger (e.g., classifier stacking, combining gazetteer look-up with our classifier etc.), but we were not able to obtain results that are significantly better than those obtained by the look-up tagger.

Finally, for the generic tagger, the results are quite low (around 33% average F-score) and in fact much lower than the results reported for this tagger when testing on newspaper texts (around 70% average F-Score). This provides evidence that our hypothesis that a generic named entity tagger is not very suitable to this type of data, is correct. This seems to apply even to non-domain-specific categories, such as PERSON and LOCATION. This is probably due to domain-specific difference between the training set for this tagger, which came from a news domain, and the test set, which came from a natural history domain. Furthermore the language used in databases is also different from that used in carefully written newspaper articles, for example, the former contains more fragmented utterances and fewer complete sentences. It is possible that these differences could be partially overcome by using a named entity tagger that was trained on *spoken* language, as spoken language may be more similar to the language used in textual databases.

3.1.3 Summary

We investigated whether it is possible to bootstrap a named entity tagger for textual databases by exploiting the fact that such database typically contain several entity-specific fields. These fields can be used to automatically extract gazetteer lists, which can then be used to obtain training material for a named entity tagger. For our database, we found that this strategy is partially useful. There is a lot of overlap between named entities in the entity-specific fields and in the free text fields. Hence, a simple gazetteer look-up leads to a fairly high average F-Score of 76%. Using the gazetteer lists to bootstrap a database trained named entity tagger was less successful. While the tagger still achieved a reasonable performance, its average F-Score (69%) was significantly below that obtained by the look-up tagger. We believe that the main problem lies in the relatively small training set. A better

strategy for future work might thus be to train on external data from the natural history domain. This could be extracted from the web by using the gazetteer lists to automatically generate suitable queries or it could be extracted from scientific papers published by Naturalis researchers or other experts in the field. Finally, we found evidence that generic named entity taggers may not be very suitable for our type of data.

3.2 Automatic Data Completion

The second enrichment task we addressed concerned the automatic completion of incomplete data. Entries with some empty cells are fairly frequent in our database, with more than 60% of all cells being unfilled. Not every unfilled cell represents an incomplete record; some fields, such as BIJZONDERHEDEN (*special remarks*) and SUB-SPECIES, do not always have to be filled. However, other fields like BIOTOOP (*biotope*) or PLAATS (*finding place*) should, ideally, be filled for all entries.

In addition to unfilled cells, there are also cells which only contain a back reference to the content of other cells, such as *idem*, as in:

VERZAMELAAR: *M.S. Hoogmoed* DONATOR: *idem*

This applies to just over 2,000 cells. Resolving these back references is not trivial. First, it is not always clear in which direction (i.e. horizontally or vertically) the reference has to be resolved; in some cases *idem* seems to refer to the content of the same field in the previous entry (i.e., vertical resolution), in other cases *idem* seems to refer to the previously filled cell in the current entry (i.e., horizontal resolution). Second, even if the resolution direction is known, changes in the database structure may mean that it cannot be guaranteed that the correct referent is found. For example, vertical back references depend on the fact that the entries are always displayed in the same order. However, this is not necessarily the case, particularly if the database is exported to another format. Similarly, horizontal back references can easily be broken by the addition of new columns. Consequently, some heuristics are needed to replace *idem* with the appropriate value for that cell. One way to do this, is by employing automatic data completion.

3.2.1 Predicting Field Contents

To fill empty cells, we exploited the fact that there are often interdependencies between different database columns. In our database, for example, there is an interdependency between the LOCATIE (*location*) and the LAND (*country*) columns: the probability that the latter contains the value *Zuid Afrika* increases if the former contains the string *Tafel Berg* (and vice versa). Similar interdependencies hold between other columns, such as LOCATIE and HOOGTE (*altitude*), LAND and BIOTOOP, or between the columns encoding a specimen's position in the zoological taxonomy (e.g., SPECIES and FAMILIE). Given enough data, these interdependencies can be learnt automatically and used to predict the value of a cell given the values of the other cells of that entry.

We tested the feasibility of this approach by training 34 TiMBL [Daelemans et al., 2004] classifiers, i.e., one for each non-empty column,⁷ to predict the value of that column for a given entry. Under this set-up, the instances to be classified are the individual entries (or more specifically the entries for which the column is filled), the class to be predicted is the content that the column has for a given entry, and the features are the contents of all other columns for that entry. The training and test

⁷We excluded the S_GUID column, which forms the primary key for the database and thus should be unique for each entry.

sets for each classifier were obtained automatically from the database. We used 90% of the entries for training and 10% for testing.

We trained the classifiers using TiMBL’s default settings⁸ and then applied them to the test sets. The prediction accuracies are shown in Table 4, together with information about the data set sizes, about the number of different values (*Values*) in a column, and about the frequency of the most frequent value (*Max. Val. Frequ.*). We also calculated two baselines. For the first (*Rand.*), we calculated the average accuracy one would expect if a value was predicted by randomly choosing from the permissible values for the column. For the second baseline (*Maj.*), we calculated the expected accuracy of always predicting the majority value.

It can be seen that the classifiers perform remarkably well, even on the “free text” columns, such as BIJZONDERHEDEN (*special remarks*, row 30), where over 76% of the 2537 possible values are predicted correctly. This is well above the baselines, which are 4% for predicting the majority value and 0.03% for predicting a random value. Similar performance levels are obtain for other fields. The highest accuracy is achieved for KLASSE (99.94%, see line 1), however this field is relatively easy to predict as it only has three values (*reptilia*, *amphibia*, and a comment about an inconsistency in the taxonomic information for one of the entries). The other taxonomic fields are also fairly well predicted, with accuracies between 88.65% (SPECIES) and 98.84% (ORDE). Prediction accuracies below 50% are only seen for fields containing data entry information, i.e., RECORDER TIME (3.92%, row 34) and REG NUMMER (0.37%, row 31). This is not surprising as these columns should be virtually impossible to predict. For example, the time at which an entry was added to the database is unlikely to be correlated to any of the other columns and the registration number (REG NUMMER) is virtually unique for any given entry (with 16,769 different values for 16,870 entries). The relatively high prediction accuracies for all other fields suggest that there are indeed many interdependencies between columns in the database. There is also a lot of repetition, even in the free text fields, i.e., many values and value combinations occur repeatedly.

3.2.2 Summary

We tested whether it is possible to predict the value of a cell based on the values of other cells in an entry. We found that most fields can be predicted with an accuracy of 56% and higher, with many fields being predictable with an accuracy in the high eighties or even nineties. This suggests that there are many interdependencies between different columns in the database and that there is also a certain amount of repetition, so that these interdependencies can be learnt automatically. This is likely to be the case for many cultural heritage databases. To test this for our domain, we plan to apply the method to other specimen databases.

4 Data Cleaning

Data cleaning refers to a variety of normalising and error correction tasks. Normalising is beneficial if the data contain, for example, proper names or dates in different formats (e.g., *M. S. Hoogmoed* vs. *Hoogmoed, M. S.*) but normalising can also mean standardising the spelling of proper names, e.g., in case of cross-linguistic variation (*Carl Linnaeus* vs. *Carolus Linnaeus* vs. *Carl von Linné*) (cf. Pouliquen et al. [2005]). Error correction can mean the detection and correction of typos (e.g. *Hoogfmoed* vs. *Hoogmoed*) but it can also refer to more serious errors and inconsistencies in the

⁸The default settings are: Algorithm: IB1, Similarity Metric: Weighted Overlap, Feature Weighting: Information Gain Ratio, $k=1$.

		Instances		Values	Max.	Accuracies (%)		
		Training	Test		Val. Frequ.	TiMBL	Maj.	Rand.
1	KLASSE	14,177	1,576	3	8,821	99.94	56.00	33.33
2	ORDE	14,174	1,575	14	6,564	98.84	44.76	10.00
3	FAMILIE	14,120	1,569	83	1,950	97.37	18.59	1.92
4	GENUS	15,012	1,668	649	1,316	91.95	10.13	0.35
5	SPECIES	14,986	1,666	1,351	1,029	88.65	6.18	0.07
6	SUB SPECIES	2,968	330	285	347	94.87	10.52	0.35
7	AUTEUR	13,538	1,505	1,057	1981	88.07	13.17	0.09
8	PUBLICATIE	2,032	226	86	1,157	97.53	51.24	1.16
9	DETERMINATOR	9,032	1,004	152	2,969	96.66	29.58	0.66
10	DETERMINATIE DATUM	2,188	244	257	507	87.67	20.85	0.39
11	TYPE	461	52	29	172	74.58	33.53	3.45
12	TYPE-NAAM	336	38	122	91	82.50	24.33	0.82
13	BIOTOOP	1,768	197	700	91	63.02	4.63	0.14
14	COORDINATEN	511	57	118	349	88.14	61.44	0.85
15	HOOGTE	2,185	243	333	145	77.48	5.97	0.300
16	LAND	1,376	153	71	529	95.06	34.60	1.41
17	PLAATS	13,597	1,511	3,566	349	74.98	2.31	0.02
18	PROVINCIE/STAAT	8,114	902	515	923	88.09	10,24	0.19
19	LAND ID	14,850	1,651	124	3,741	95.49	22.67	0.81
20	LOCATIE	1,398	156	653	349	67.66	22.46	0.15
21	VERZAMELAAR	13,458	1,496	1,079	4,552	85.25	30.44	0.09
22	VERZAMELNUMMER	7,404	823	4,095	116	56.49	1.41	0.02
23	VERZAMELDATUM	12,859	1,429	3,240	263	65.50	1.84	0.03
24	AANTAL	13,794	1,533	39	14,521	93.33	94.74	2.56
25	SEXE	2,652	295	49	1,557	64.86	52.83	2.04
26	BEWAARMETHODE	13,833	1,537	47	14,945	98.93	97.23	2.13
27	DONATOR	3,955	440	523	2,007	87.83	45,67	0.19
28	INGEKOMEN	8,229	915	811	2,128	88.98	23.27	0.12
29	ETIKET GEGEVENS	7,915	880	1,813	116	93.37	1.32	0.05
30	BIJZONDERHEDEN	8,649	962	2537	391	76.21	4.07	0.03
31	REG NUMMER	15,183	1,687	16,769	10	0.37	0.06	<0.01
32	RECORDER	15,183	1,687	10	9,489	98.12	56,25	10.00
33	RECORDER DATE	15,166	1,686	534	1,340	—	7.95	0.19
34	RECORDER TIME	8,997	1000	7,554	25	3.92	0.25	0.01

Table 4: Predicting Cell Values: Number of instances in the training and test sets (*Instances Training*, *Instances Testing*); Number of different values (*Values*); Frequency of the most frequent value (*Max. Val. Frequ.*); Prediction accuracies for TiMBL and two Baselines: predicting the majority value (*Maj.*) and predicting a random value (*Rand.*). No TiMBL accuracy is given for RECORDER DATE (line 33) because the values for this field were corrupted when we ran the experiments.

data, such as a wrong country name in a field book or database entry, or an erroneous taxonomic classification.

Usually it is impossible to avoid errors completely, even in well maintained databases. Errors can arise for a variety of reasons, ranging from technical limitations (e.g., copy-and-paste errors) to different interpretations of what type of information should be entered into different database fields. The latter situation is especially prevalent if the database is maintained by several people. Manual identification and correction of errors is very time consuming, even for relatively small databases, and thus usually infeasible. A more realistic approach uses automatic means to identify potential errors; these can then be flagged to a human expert, and subsequently corrected manually or semi-automatically.

While there has been a significant amount of previous research on identifying and correcting errors in data sets, most methods are not particularly suited for textual databases (see Section 4.1). We developed two techniques that aim to fill this gap. Both are data-driven and knowledge-lean; errors are identified through comparisons with other database fields. The first method compares the database fields of individual entries and flags improbable combinations as potential errors. Because the focus is on individual entries, i.e., *rows* in the database, we call this *horizontal error correction*. The second method aims at a different type of error, namely values which were entered in the wrong *column* of the database. Potential errors of this type are determined by comparing the content of a database cell to all database columns and determining which column it fits best. Because the focus is on columns, we refer to this method as *vertical error correction*.

4.1 Related Work on Error Correction

There is a considerable body of previous work on the generic issue of data cleaning. Much of the research directed specifically at databases focuses on identifying identical records when two databases are merged [Hernández and Stolfo, 1998, Galhardas et al., 1999]. This is a non-trivial problem as records of the same objects coming from different sources typically differ in their primary keys. There may also be subtle differences in other database fields. For example, names may be entered in different formats (e.g., *John Smith* vs. *Smith, J.*) or there may be typos which make it difficult to match fields (e.g., *John Smith* vs. *Jon Smith*).⁹

In a wider context, a lot of research has been dedicated to the identification of outliers in datasets. Various strategies have been proposed. The earliest work uses probability distributions to model the data; all instances which deviate too much from the distributions are flagged as outliers [Hawkins, 1980]. This approach is called *distribution-based*. In *clustering-based* methods, a clustering algorithm is applied to the data and instances which cannot be grouped under any cluster, or clusters which only contain very few instances are assumed to be outliers (e.g., Jiang et al. [2001]). *Depth-based* methods (e.g., Ruts and Rousseeuw [1996]) use some definition of depth to organise instances in layers in the data space; outliers are assumed to occupy shallow layers. *Distance-based* methods [Knorr and Ng, 1998] utilise a k -nearest neighbour approach where outliers are defined, for example, as those instances whose distance to their nearest neighbour exceeds a certain threshold. Finally, Marcus and Maletic [2000] propose a method which learns association rules for the data; records that do not conform to any rules are then assumed to be potential outliers.

In principle, techniques developed to detect outliers can be applied to databases as well, for instance to identify cell values that are exceptional in the context of other values in a given column, or to identify database entries that seem unlikely compared to other entries. However, most methods

⁹The problem of whether two proper noun phrases refer to the same entity has also received attention outside the database community [Bagga, 1998].

are not particularly suited for textual databases. Some approaches only work with numeric data (e.g., distribution-based methods), others can deal with categorical data (e.g., distance-based methods) but treat all database fields as atoms. For databases with free text fields it can be fruitful to look at individual tokens within a text string. For instance, units of measurement (*m*, *ft*, etc.) may be very common in one column (such as HOOGTE (altitude)) but may indicate an error when they occur in another column (such as VERZAMELAAR (collector)).

4.2 Horizontal Error Correction

In Section 3.2, we found that the content of a field can often be predicted to a high accuracy by taking into account the other fields of that entry. This is due to the fact that the different fields in a database are often not statistically independent; i.e., for a given entry, the likelihood of a particular value in one field may be affected by the values in (some of) the other fields. In this section, we explore whether this can be exploited to detect database fields that are inconsistent with the values of the other cells and thus are likely to be erroneous. The main idea is to flag as potential errors all cells for which the predicted value deviates from the original value in the database. This approach bears some similarity to Marcus and Maletic's [2000] method of inferring association rules and then looking for outliers relative to these rules. However, we do not explicitly infer rules.

We applied the method to the four highest taxonomic fields in our database (KLASSE, ORDE, FAMILIE and GENUS), because it is possible, albeit somewhat time-consuming, for a non-expert to check the values of these fields against a published zoological taxonomy. Strictly speaking, it is only possible to check for *consistency* (within an entry and with the published taxonomy). For example, if the genus field contains the value *Duellmanohyla*, the species field contains *californicus* and the published taxonomy does not list *californicus* as a possible species under the genus *Duellmanohyla*, this points to an inconsistency in the database entry (i.e., the value of the genus field is not consistent with that of the species field). If the family field for the entry has the value *Bufo*, one can also hypothesise that the error stems from the genus field (because *Duellmanohyla* is not a possible genus for the family *Bufo*) and not from the species field (because *californicus* is a possible species in the family *Bufo*); one can also hypothesise that the correct value for the genus field is *Bufo*, because this value resolves the inconsistency in the entry. However, there is no guarantee that the specimen to which the entry refers is indeed a *Bufo californicus*, as it is possible (though less likely) that the value in the genus field is correct and both the order and the species fields are wrong. Similarly, there is also a remote possibility that the values of *all* taxonomic fields are wrong, e.g., because the container in which the specimen is stored was accidentally swapped with that of a completely different specimen. Hence, without expert knowledge and access to the actual specimens it is impossible to know for sure what the “correct” value for a given taxonomic field is (i.e., the value which is correct for the specimen to which the entry refers). However, it is possible to make an educated guess about the correctness of a field based on its consistency with other fields; this is the strategy we employed when assessing whether an error flagged by our automatic error detection method was likely to be a real error or not.

While such a consistency check works relatively well for the four highest taxonomic fields, it works less well for the two lowest fields: SPECIES and SUB SPECIES. These two fields contain the most specific information for a database entry.¹⁰ This means that, while an inconsistency between their value and the other taxonomic fields points to a possible error, the reverse is not true; even if the value of the species field is consistent with the other fields there is still a significant probability

¹⁰The SUB SPECIES field is rarely filled; hence, for most entries, SPECIES is the most specific field.

that it is wrong because there are often several possible species for a given genus, and —because SPECIES and SUB SPECIES are the most specific taxonomic fields— there is no further evidence for the (likely) correctness of a value.¹¹ Therefore, we decided to exclude SPECIES and SUB SPECIES from the evaluation.

We split the data into 80% training set, 10% development set and 10% test set. As not all taxonomic fields are filled for all entries, the exact sizes for each data set differ, depending on which field is to be predicted (see Table 5).

We used the development data to set TiMBL’s parameters, such as the number of nearest neighbours to be taken into account or the similarity metric [van den Bosch, 2004]. Ideally, one would want to choose the setting which optimised the *error detection accuracy*. However, this would require manual annotation of the errors in the development set. As this is fairly time consuming, we abstained from it. Instead we chose the parameter setting which maximised the *value prediction accuracy* for each taxonomic field, i.e. the setting for which the disagreement between the values predicted by TiMBL and the values in the database was smallest. The motivation for this was that a high prediction accuracy will minimise the number of potential errors that get flagged (i.e., disagreements between TiMBL and the database) and thus, hopefully, lead to a higher error detection precision, i.e., less work for the human annotator who has to check the potential errors.

	training	devel.	test
CLASS	7,495	937	937
ORDER	7,493	937	937
FAMILY	7,425	928	928
GENUS	7,891	986	986

Table 5: Data set sizes for different taxonomic fields

We also used the development data to perform some feature selection. We compared (i) using the values of all other fields (for a given entry) as features and (ii) only using the other taxonomic fields plus the author field, which encodes which taxonomist first described the species to which a given specimen belongs.¹² The reduced feature set was found to lead to better or equal performance for all taxonomic fields and was thus used in the experiments reported below.

To determine whether this method is suitable for semi-automatic error correction, we looked at the cases in which the value predicted by TiMBL differed from the original value. There are three potential reasons for such a disagreement: (i) the value predicted by TiMBL is wrong, (ii) the value predicted by TiMBL is correct and the original value in the database is wrong, and (iii) both values are correct and the two terms are (zoological) synonyms. We checked the values predicted by TiMBL against two published zoological taxonomies¹³ and counted how many times the predicted value was the correct value. Because the focus is on error *correction* rather than error *detection*, cases where both the value predicted by TiMBL and the original value in the database were wrong, were counted as TiMBL errors.

¹¹In some cases the author field might provided such evidence. However, taking this field into account as well, would have made the manual consistency check much more time consuming.

¹²The author information provides useful cues for the prediction of taxonomic fields because taxonomists often specialise on a particular zoological group. For example, a taxonomist who specialises on *Ranidae* (frogs) is unlikely to have published a description of a species belonging to *Serpentes* (snakes).

¹³We used the ITIS Catalogue of Life (<http://www.species2000.org/2005/search.php>) and the EMBL Reptile Database (<http://www.embl-heidelberg.de/~uetz/LivingReptiles.html>).

	prediction acc.	disagreements	database errors	synonyms	TiMBL errors
CLASS	99.87%	2	50.00% (1)	0% (0)	50.00% (1)
ORDER	98.29%	26	38.00% (10)	19.00% (5)	43.00% (11)
FAMILY	98.02%	33	9.09% (3)	36.36% (12)	54.55% (18)
GENUS	92.57%	135	5.93% (8)	4.44% (6)	89.63% (121)

Table 6: Error correction precision (horizontal method), showing the accuracy with which cell contents can be predicted (*prediction acc.*), the percentage and absolute number of (i) the disagreements between the TiMBL classifier and the database, (ii) the taxonomic synonyms, and (iii) the TiMBL errors

Table 6 shows the results (the absolute numbers of database errors, synonyms and TiMBL errors are shown in brackets). It can be seen that TiMBL detects several errors in the database and predicts the correct values for them. It also finds several synonyms. For GENUS, however, the vast majority of disagreements between TiMBL and the database is due to TiMBL errors. This can be explained by the fact that GENUS is relatively low in the taxonomy (directly above SPECIES). As the values of higher fields only provide limited cues for the value of a lower field, the lower a field is in the taxonomy the more difficult it is to predict its value accurately.

So far we have only looked at the *precision* of our error detection method. Error detection *recall* is often difficult to determine precisely because this would involve manually checking the dataset (or a significant subset) for errors, which is typically quite time-consuming. As an alternative, we estimated recall by introducing errors artificially and determining what percentage of these artificial errors was detected. For each taxonomic field, we changed the value of 10% of the entries, which were randomly selected. In these entries, the original values were replaced by one of the other attested values for this field. The new value was selected randomly and with 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, e.g., some values may be more easily confused by humans than others. Table 7 shows the results. The estimated recall is fairly high; in all cases above 90%. This suggests that a significant proportion of the errors is detected by our method.

	recall
CLASS	95.56%
ORDER	96.82%
FAMILY	96.15%
GENUS	93.09%

Table 7: Recall for artificially introduced errors (horizontal method)

4.3 Vertical Error Detection

While the horizontal method described in the previous section aimed at correcting values which are inconsistent with the remaining fields of a database entry, vertical error correction is aimed at a different type of error, namely, text strings which were entered in the wrong column of the database. For example, in our database, information about the biotope in which a specimen was found may have been entered in the BIJZONDERHEDEN (special remarks) column rather than the BIOTOOP (biotope)

column. Errors of this type are quite frequent. They can be accidental, i.e., the person entering the information inadvertently chose the wrong column, but they can also be due to misinterpretation, e.g., the person entering the information may believe that it fits the BIJZONDERHEDEN column better than the BIOTOOP column or they may not know that there is a BIOTOOP column. Some of these errors may also stem from changes in the database structure itself, e.g., maybe the BIOTOOP column was only added after the data was entered.

Identifying this type of error can be recast as a text classification task: given the content of a cell, i.e., a string of text, the aim is to determine which column the string most likely belongs to. Text strings which are classified as belonging to a different column than they are currently in, represent a potential error. Recasting error detection as a text classification problem allows the use of supervised machine learning methods, as training data (i.e., text strings labelled with the column they belong to) can easily be obtained from the database.

We tokenised the text strings in all database fields and labelled them with the column they occur in. Each string was represented as a vector of 48 features, encoding the (i) string itself and some of its typographical properties (13 features), and (ii) its similarity with each of the 35 columns (in terms of weighted token overlap) (35 features).

The typographical properties we encoded were: the number of tokens in the string and whether it contained an initial (i.e., an individual capitalised letter), a number, a unit of measurement (e.g., *km*), punctuation, an abbreviation, a word (as opposed to only numbers, punctuation etc.), a capitalised word, a non-capitalised word, a short word (< 4 characters), a long word, or a complex word (e.g., containing a hyphen).

The similarity between a string, consisting of a set T of tokens $t_1 \dots t_n$, and a column col_x was defined as:

$$sim(T, col_x) = \frac{\sum_{i=1}^n t_i \times tfidf_{t_i, col_x}}{|T|}$$

where $tfidf_{t_i, col_x}$ is the tfidf weight (*term frequency - inverse document frequency*, cf. Sparck-Jones [1972]) of token t_i in column col_x . This weight encodes how representative a token is of a column. The term frequency, tf_{t_i, col_x} , of a token t_i in column col_x is the number of occurrences of t_i in col_x divided by the number of occurrences of all tokens in col_x . The term frequency is 0 if the token does not occur in the column. The inverse document frequency, idf_{t_i} , of a token t_i is the number of all columns in the database divided by the number of columns containing t_i . Finally, the tfidf weight for a term t_i in column col_x is defined as:

$$tfidf_{t_i, col_x} = tf_{t_i, col_x} \log idf_{t_i}$$

A high tfidf weight for a given token in a given column means that the token frequently occurs in that column but rarely in other columns, thus the token is a good indicator for that column. Typically tfidf weights are only calculated for content words, however we calculated them for all tokens, partly because the use of stop word lists to filter out function words would have jeopardised the language independence of our method and partly because function words and even punctuation can be very useful for distinguishing different columns. For example, prepositions such as *under* often indicate BIOTOOP, as in *under a stone*.

To assign a text string to one of the 35 non-empty database columns, we trained TiMBL [Daelmans et al., 2004] (with the default setting) on the feature vectors of all other database cells labelled with the column they belong to. Cases where the predicted column differed from the current column of the string were recorded as potential errors.

We applied the classifier to all filled database cells. For each of the strings identified as potential errors, we checked manually (i) whether this was a real error and (ii) whether the column predicted by

the classifier was the correct one. While checking for this type of error is much faster than checking for errors in the taxonomic fields, it is sometimes difficult to tell whether a flagged error is a real error. In some cases it is not obvious which column a string belongs to, for example because two columns are very similar in content (such as LOCATIE (location) and PLAATS (finding place)), in other cases the content of a database field contains several pieces of information which would best be located in different columns. For instance, the string *found with broken neck near Karlobag* arguably could be split between the BIJZONDERHEDEN (special remarks) and the LOCATIE columns. We were conservative in the first case, i.e., we did not count an error as correctly identified if the string could belong to the original column, but we gave the algorithm credit for potential errors where part of the string should be in a different column.

The results are shown in the second column (*unfiltered*) in Table 8. The classifier found 836 potential errors, 148 of these were found to be real errors. For 100 of the correctly identified errors the predicted column was the correct column. Some of the corrected errors can be found in Table 9. Note that the system corrected errors in both English and Dutch text strings without requiring language identification or any language-specific resources (apart from tokenisation).

	unfiltered	filtered
flagged errors	836	262
real errors	148	67
correctly corrected	100	54
precision error detection	17.70 %	25.57%
accuracy error correction	67.57%	80.60%

Table 8: Results automatic error detection and correction for all database fields (vertical method)

We also calculated the precision of error detection (i.e., the number of real errors divided by the number of flagged errors) and the error correction accuracy (i.e., the number of correctly corrected errors divided by the number correctly identified errors). The error detection precision is relatively low (17.70%). In general a low precision means relatively more work for the human expert checking the flagged errors. However note that the system considerably reduces the number of database fields that have to be checked (i.e., 836 out of 229,430 filled fields). We also found that, for this type of error, error checking can be done relatively quickly even by a non-expert; checking the 836 errors took less than 30 minutes. Furthermore, the correction accuracy is fairly high, i.e., for most of the correctly identified errors the correct column is suggested. This means that for most errors the user can simply choose the column suggested by the classifier.

In an attempt to increase the detection precision we applied a couple of filters and only flagged errors which passed these filters. First, we filtered out potential errors if the original and the predicted column were of a similar type (e.g., if both contained person names or dates) as we noticed that our method was very prone to misclassifications in these cases.¹⁴ For example, if the name *M.S. Hoogmoed* occurs several times in the VERZAMELAAR (collector) column and a few times in the DONATOR column, the latter cases are flagged by the system as potential errors. However, it is entirely normal for a person to occur in both the VERZAMELAAR and the DONATOR column. What is more, it is impossible to determine on the basis of the text string *M.S. Hoogmoed* alone, whether the correct column for this string in a given entry is DONATOR or VERZAMELAAR or both.¹⁵ Secondly, we only

¹⁴Note, that this filter requires a (very limited) amount of background knowledge, i.e. knowledge about which columns are of a similar type.

¹⁵Note, however, that the horizontal error detection method proposed in the previous section might detect an erroneous

string	original column	corrected column
op boom ongeveer 2,5 m boven grond (<i>on a tree about 2.5 m above ground</i>)	SPECIAL REMARKS	BIOTOPE
25 km N.N.W Antalya	SPECIAL REMARKS	LOCATION
1700 M	BIOTOPE	ALTITUDE
gestorven in gevangenschap 23 september 1994 (<i>died in captivity 23 September 1994</i>)	LOCATION	SPECIAL REMARKS
roadside bordering secondary forest	LOCATION	BIOTOPE
Suriname Exp. 1970 (<i>Surinam Expedition 1970</i>)	COLLECTION NUMBER	COLLECTOR

Table 9: Examples of automatically corrected errors (vertical method)

flagged errors where the predicted column was empty for the current database entry. If the predicted column is already occupied, the string is unlikely to belong to that column (unless the string in that column is also an error). The third column in Table 8 (*filtered*) shows the results. It can be seen that detection precision increases to 25.57% and correction precision to 80.60%, however the system also finds noticeably fewer errors (67 vs. 148).

	unfiltered		filtered	
	Prec.	Rec.	Prec.	Rec.
BIO.	13.55%	42.00%	1.39%	2.00%
PUB.	10.35%	100.00%	13.04%	75.00%
BIJZ.	10.06%	16.50%	12.31%	8.00%

Table 10: Precision and Recall for three free text columns (vertical method)

Estimating the error detection recall (i.e., the number of identified errors divided by the overall number of errors in the database) would involve manually identifying all the errors in the database. This was not feasible for the database as a whole. Instead we manually checked three of the free text columns, namely, BIOTOOP (BIO.), PUBLICATIE (PUB.) and BIJZONDERHEDEN (special remarks, BIJZ.), for errors and calculated the recall and precision for these. Table 10 shows the results. In the unfiltered case the recall is fairly high for PUBLICATIE, not so high for BIOTOOP, and relatively low for the BIJZONDERHEDEN. This is probably due to the fact that this column is very heterogeneous, thus it is fairly difficult to find the true errors in it. As expected, filtering increases the precision and decreases the recall, with the exception of BIOTOOP, where both values decrease significantly. The reason for this is probably that both filters are affected by the error correction accuracy, i.e., the better the correct column for an error can be predicted the better the filters work. While the correction accuracy is relatively high for most columns (see Table 8), it is fairly low for BIOTOOP, as this column is very similar to several other columns in the database (e.g., LOCATIE (location), PLAATS (finding place)). This makes it difficult to automatically determine the correct target column for an error in the BIOTOOP column.

occurrence of this string (based on the values of other fields in the entry).

4.4 Summary

We have presented two methods for (semi-)automatic error detection and correction in textual databases. The two methods are aimed at different types of errors: *horizontal error correction* attempts to identify and correct inconsistent values within a database record; *vertical error correction* is aimed at values which were accidentally entered in the wrong column. Both methods are data-driven and require little or no background knowledge. The methods are also language-independent and can be applied to multi-lingual databases. While we utilise supervised machine learning, no manual annotation of training data is required, as the training set is obtained directly from the database.

We tested the two methods on an animal specimens database and found that a significant proportion of errors could be detected: up to 97% for horizontal error detection and up to 100% for vertical error detection. While the error detection precision was fairly low for both methods (up to 55% for the horizontal method and up to 25.57% for the vertical method), the number of potential errors flagged was still sufficiently small to check manually. Furthermore, the automatically predicted correction for an error was often the right one. Hence, it would be feasible to employ the two methods in a semi-automatic error correction set-up where potential errors together with a suggested correction are flagged and presented to a user.

As the two error correction methods are to some extent complementary, it would be worthwhile to investigate whether they can be combined. Some errors flagged by the horizontal method will not be detected by the vertical method, for instance, values which are valid in a given column, but inconsistent with the values of other fields. On the other hand, values which were entered in the wrong column should, in theory, also be detected by the horizontal method. For example, if the correct FAMILY for *Rana aurora* is *Ranidae*, it should make no difference whether the (incorrect) value in the FAMILY field is *Bufo*, which is a valid value for FAMILY but the wrong family for *Rana aurora*, or *Amphibia*, which is not a valid value for FAMILY but the correct CLASS value for *Rana aurora*; in both cases the error should be detected. Hence, if both methods predict an error in a given field this should increase the likelihood that there is indeed an error. This could be exploited to obtain a higher precision. We plan to experiment with this idea in future research.

5 Conclusion

In the pilot study of the MITCH project, we looked at three data clean-up and enrichment tasks: named-entity tagging, data completion and error correction. We developed a set of data-driven techniques for these tasks and tested them on a database containing information about animal specimens in Naturalis's collection. For all three tasks we obtained reasonable results, despite the fact that none of our methods makes use of manually annotated training material or extensive background knowledge. We also largely avoided the use of language-specific processing tools; the only such tool we used was a tokeniser for Dutch. Because our methods are knowledge-lean, we believe that they can also be applied to other textual databases from the cultural heritage domain and we will test them on the two other specimens databases we have currently available.

5.1 Ongoing and Future Work

Over the next few months we will continue to work on the specimens databases. We plan to concentrate most effort on the error correction task, as we believe that this is crucial for successful data mining and information extraction. It also seems that previous research on error correction has largely ignored textual databases. Given the prevalence of this type of database, especially in the cultural

heritage domain, this is an important task to address. Among the improvements we plan to make over the next few months are:

Vertical Error Correction Our error detection method is aimed at determining whether the value of a database cell is in the wrong column. However, we found that many cells contain two pieces of information which should be in different columns. For example, in the string *found with broken neck near Karlobag*, the substring *found with broken neck* should be in the special remarks column and the substring *near Karlobag* should be in the location column. Currently, our error detection method has no means of breaking up a text string and placing it in two different columns. One way to address this would be by treating vertical error correction as a sequence labelling task, similar to named entity tagging. Under this set-up, each token in a string would be assigned a tag encoding which column the token in the given context is most likely to belong to. For example, the string *found with broken neck near Karlobag* could be tagged as:

found_SR with_SR broken_SR neck_SR near_LOC Karlobag_LOC

where the tag SR indicated the SPECIAL REMARKS column and LOC indicates the location column. Training data for this task could be generated automatically from the database. One advantage of this approach would be that it could also be used to create a database automatically from field book entries, i.e., such a tagger could be used to determine which elements of the entry should be in which column of the database.

Horizontal Error Correction Currently, our horizontal error correction method treats text strings as atoms. It might be possible to improve results by adding features that allow the classifier to 'look inside' individual text strings, similarly to the way the vertical error correction does.

Hierarchical Error Correction We also plan to explore a third error correction strategy. This strategy would be aimed at identifying errors in the taxonomic fields. The main idea is to infer a gold standard taxonomic hierarchy from the database and then use this to correct fields which seem to be inconsistent with the hierarchy. Of course, theoretically, the same could be done with a published taxonomy. However, we found that, while taxonomic information is often provided in the form of searchable websites on the internet, digitised taxonomies are frequently treated as proprietary data and not distributed freely in a format that would allow automatic processing. Inferring a taxonomy directly from our database would provide a way around this. The task is not trivial, however, due to a number of factors. First, biological taxonomies are prone to change, especially at the lower levels. For instance, advances in research can mean that a species is re-classified as belonging to a genus different from the one it was originally grouped under. Hence, some taxonomic information in our database might be outdated. A second problem is that not all taxonomic names are unique. For example, species and sub species names are often re-used in different branches of the hierarchy. For example, the two species *Iphisa elegans* and *Thamnophis elegans* both share the same species term (*elegans*) but belong to two different genera (*Iphisa* and *Thamnophis*). Hence, the fact that a species seems to be grouped under two different genera does not necessarily point to an error in the database, though in some case it could. We will investigate whether additional information (e.g. about the author of a species description) and an intelligent use of frequency information could provide a way around these problems.

Named Entity Tagging We are currently also exploring whether it is feasible to automatically generate multi-lingual gazetteer lists for geographical names by combining two bootstrapping loops: one language internal and one cross-linguistic. Better gazetteer lists should improve named entity tagging results for our database, given the high performance of the look-up tagger. We focus on geographical names because for all taggers LOCATION was the most difficult entity category to predict. Having multi-lingual gazetteer lists would also allow us to normalise geographical names to one language.

Combined Techniques Over the next few months, we will also put more weight on combining the different techniques we developed. For example, instead of correcting potential errors with respect to a particular error correction strategy, it might be beneficial to take a more holistic view by combining different strategies and correcting errors in such a way that the probability of the database as a whole is maximised. It might also be useful to employ named entity tagging as a preprocessing step for error correction, as entity information might provide useful cues about potential errors. Similarly, we will also explore whether the use of language-specific tools improves performance levels. In that case it would be possible to use automatic language identification to select the appropriate tools in cases where such tools are available but back off to the knowledge-lean method in cases where no such tools are available for the language.

Interface Design and User Study Now that we have some first results and implementations of tools that we hope will be useful for staff at Naturalis, we will also be looking into how to tailor our tools to the needs of biological researchers. At the moment this applies particularly to error correction, which is best done semi-automatically. For this to work well, the development of an easy to use interface is crucial. An initial investigation into which types of error correction interfaces work best in a given situation, will be carried out during the first half of 2006 as part of a Master's thesis by Niels Bosman, who is currently studying at Tilburg University. This research will involve gathering requirements for the main user group (i.e., researchers and curators at Naturalis), designing and implementing different interface prototypes, and empirically evaluating them in a user study. The evaluation will look at factors such as, how easy is it to use a particular interface (i.e., user-satisfaction), and how successful are the users in completing the error correction task (i.e., with respect to accuracy and speed). A number of biologists at Naturalis have agreed to support this study by devoting time to help us gathering specifications and by testing the resulting interfaces. We are also looking into the possibility of recruiting a number of biology students for the experiments. This would allow us to compare different user groups (senior researchers and curators vs. biology students). We will build on this initial work and carry out further user studies throughout the MITCH project.

5.2 Deliverables

We have presented selected results of the pilot project in oral and poster presentations. We also made three submissions to international conferences and workshops. Two of these have been accepted in the meantime; a decision about the third is expected soon. We are currently working on further submissions. In detail, the research output so far has been as follows:

- **peer-reviewed publications:**
 - a paper comparing the two error correction methods has been accepted for the EACL workshop on “Adaptive Text Extraction and Mining (ATEM)” to be held in Trento, Italy, 4. April 2006 (see Sporleder et al. [2006a])

- a paper reporting our findings on named entity tagging has been accepted for the Fifth International Conference on Language Resources and Evaluation (LREC-2006), 24–26 May 2006, Genoa, Italy (see Sporleder et al. [2006b])
- an abstract about the vertical error correction method is under review for Digital Humanities 2006, 5. & 6. July 2006, Paris, France.

- **posters, oral presentations, demos:**

- a poster outlining our pilot project was presented at SIREN 2005 (Scientific ICT Research Event Netherlands), 6. October 2005, Eindhoven, The Netherlands.
- a poster summarising the results of the pilot project will be presented at the Annual Conference of the German Society for Linguistics, Special Interest Group Computational Linguistics, 23. February 2006, Bielefeld, Germany.
- we presented the results of the horizontal error correction method at the 16th Meeting of Computational Linguistics in the Netherlands (CLIN), 16. December 2005, Amsterdam, The Netherlands.
- a demo of our semi-automatic error-correction methods was shown at the CATCH meeting on 20. January 2006, Den Haag, The Netherlands.

Submissions and other research outputs planned for the future include:

- **peer-reviewed publications:**

- Marieke van Erp, PhD student of the MITCH project, is currently working on a paper about the automatic acquisition of multi-lingual gazetteer lists for geographical names. This paper will be submitted to the ESLLI 2006 Student Session, 37. July–11. August, Malaga, Spain.
- we are also working on a submission to BENELEARN 2006, 11. & 12. May 2006, Ghent, Belgium.
- we intend to write a journal paper about different error correction methods for textual databases, to be submitted in autumn 2006. Possible target journals include *Language Resources and Evaluation*, *Literary and Linguistic Computing*, *Natural Language Engineering* and *Journal of Artificial Intelligence Research*.

- **other:**

- a cleaned and enriched version of the Reptiles and Amphibians database will be made available to the researchers at Naturalis soon.
- a masters thesis about different user interface for semi-automatic error correction will be written in the first half of 2006.

References

- Amit Bagga. *Coreference, Cross-Document Coreference, and Information Extraction Methodologies*. PhD thesis, Dept. of Computer Science, Duke University, 1998.
- Toine Bogers. Dutch named entity recognition: Optimizing features, algorithms, and output. Master's thesis, Tilburg University, 2004.
- Kalina Bontcheva, Diana Maynard, Hamish Cunningham, and Horacio Saggion. Using human language technology for automatic annotation and indexing of digital library content. In *Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries (ECDL-02)*, 2002.
- Sabine Buchholz and Antal van den Bosch. Integrating seed names and ngrams for a named entity list and classifier. In *Proceedings of the 2000 Conference on Language Resources and Evaluation (LREC-00)*, 2000.
- Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*, 1999.
- Silviu Cucerzan and David Yarowsky. Language independent named entity recognition combining morphological and contextual evidence. In *Proceedings of the Joint SIGDAT Conference on Empirical Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*, 1999.
- Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. *TIMBL: Tilburg Memory Based Learner, version 5.1, Reference Guide*, 2004. ILK Research Group Technical Report Series no. 04-02.
- Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. In *Proceedings of the 2003 Conference on Natural Language Learning (CoNLL-03)*, pages 168–171, 2003.
- Helena Galhardas, Daniela Florescu, Dennis Shasha, and Eric Simon. An extensible framework for data cleaning. Technical Report RR-3742, INRIA Technical Report, 1999.
- Douglas M. Hawkins. *Identification of outliers*. Chapman and Hall, London, 1980.
- Mauricio A. Hernández and Salvatore J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Journal of Data Mining and Knowledge Discovery*, 2:1–31, 1998.
- Mon-Fong Jiang, Shian-Shyong Tseng, and Chih-Ming Su. Two-phase clustering process for outliers detection. *Pattern Recognition Letters*, 22:691–700, 2001.
- Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98)*, 1998.
- Winston Lin, Roman Yangarber, and Ralph Grishman. Bootstrapped learning of semantic classes from positive and negative examples. In *Proceedings of the ICML-03 Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.

- Andrian Marcus and Jonathan I. Maletic. Utilizing association rules for identification of possible errors in data sets. Technical Report TR-CS-00-04, The University of Memphis, Division of Computer Science, 2000.
- Diana Maynard, V. Tablan, and Hamish Cunningham. NE recognition without training data on a language you don't speak. In *Proceedings of the ACL Workshop on Multilingual and Mixed-language Named Entity Recognition: Combining Statistical and Symbolic Models*, Sapporo, Japan, 2004.
- Andrei Mikheev, Marc Moens, and Claire Grover. Named entity recognition without gazetteers. In *Proceedings of EACL*, Bergen, Norway, 1999.
- D. Palmer and D. Day. A statistical profile of the named entity task. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP)*, Washington, DC, 1997.
- William Phillips and Ellen Riloff. Exploiting strong syntactic heuristics and co-training to learn semantic lexicons. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, 2002.
- Bruno Pouliquen, Ralf Steinberger, Camelia Ignat, Irina Temnikova, Wajdi Zaghouni, and Jan Žižka. Detection of person names and their translations in multilingual news. In *Colloque Tratement lexicographique des noms propres*, Tours, 24 March, 2005, 2005.
- Ida Ruts and Peter J. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23:153–168, 1996.
- Karen Sparck-Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.
- Caroline Sporleder, Marieke van Erp, Tijn Porcelijn, and Antal van den Bosch. 'Spotting the odd-one-out': Data-driven error detection and correction in textual databases. In *Proceedings of the EACL 2006 Workshop on Adaptive Text Extraction and Mining (ATEM)*, Trento, Italy, 2006a.
- Caroline Sporleder, Marieke van Erp, Tijn Porcelijn, Antal van den Bosch, and Pim Arntzen. Identifying named entities in text databases from the natural history domain. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-06)*, Genoa, Italy, 2006b.
- Erik F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 2002 Conference on Computational Natural Language Learning (CoNLL-02)*, pages 155–158, 2002.
- Antal van den Bosch. Wrapped progressive sampling search for optimizing learning algorithm parameters. In *Proceedings of the 16th Belgian-Dutch Conference on Artificial Intelligence*, pages 219–226, 2004.

Appendix: Columns in the Reptiles and Amphibians Database

Ten columns in the database are unused, namely ACTIVITEIT, ACTIVITEITSPERIODE, GEN_BIJZONDERHEDEN, S_GENERATION, S_LINEAGE, STANDPLAATS, SOORT MATERIAAL, REFERENTIENUMMER and ZON/SCHADUW. The other fields and their contents are briefly described below.

Taxonomy related information

- **KLASSE** Taxonomic name of class. 7% are empty; 4 values, but only Amphibia and Reptilia are valid
- **ORDE** Taxonomic name of the order. 7% are empty; 15 different categories of which 4 appear to be duplicates due to spelling variations
- **FAMILIE** Taxonomic name of family. 7% are empty; 83 different values with high similarities
- **GENUS** Taxonomic name of genus. Rarely empty; 638 different categories with high similarities
- **SPECIES** Taxonomic name for species. Rarely empty; 1317 different values with spelling variations and abbreviations possibly forced by database limitations
- **SUB SPECIES** Taxonomic name for subspecies. 80% are empty; 284 different values including spelling variations
- **AUTEUR** Name(s) and year of publication in taxonomy. 10% empty; 1036 values
- **PUBLICATIE** Free text field referring to/elaborating on publication in English and Dutch. 89% of the fields are empty; 84 values including spelling variations
- **DETERMINATOR** Name of determinator, sometimes also date of determination. 39% are empty; 145 different values
- **DETERMINATIE DATUM** Date of determination. Most entries are empty; mostly DD-MM-YYYY format
- **TYPE** Type characterisation of specimen. Most entries are empty; 25 different values including spelling variations
- **TYPE-NAAM** Type exemplar name. Most entries are empty; 121 different values including spelling variations; may include name of collector and year of collection

Information on the collection of the specimen

- **BIOTOOP** Description of environmental conditions at finding place. 88% are empty; 679 different values including spelling and mark-up variations; English and Dutch
- **COORDINATEN** Coordinates of finding place. Most entries are empty
- **HOOGTE** Altitude level of finding place. 85% are empty; 330 different values; different units of measurement
- **LAND** Name of country where specimen was found. Mostly empty

- PLAATS Nearest town or description of location related to nearest town. 10% are empty; 3436 different values including spelling variations
- PROVINCIE/STAAT Province or state of finding place. 46% are empty; 496 different values including spelling variations
- LAND ID Unique number referring to country where specimen was found. 1% are empty; 119 different values; also holds 'unknown' or invalid values
- LOCATIE Free text field elaborating on finding place and circumstances. 91% are empty; 655 different values including spelling variations; English and Dutch
- VERZAMELAAR Name(s) of collector(s). 10% are empty; 1056 different values including spelling variations
- VERZAMELNUMMER Non-unique identifier used by collector. 50% are empty; 4061 different values including mark-up variations
- VERZAMELDATUM Collection date. 14% are empty; 3204 different values including mark-up variations; mostly numeric and in DD-MM-YYYY format
- VERZAMELDATUM OUD Collection date in old format. Most entries are empty; 81 different values including spelling and mark-up variations

Information on conservation

- AANTAL Number of specimens the record refers to. 86 % of the records refer to 1 specimen
- SEXE Description of sex, including 'juvenile'. 81% are empty; 44 different values including spelling variations
- BEWAARMETHODE Description of conservation method. 9% are empty; 41 different values including spelling variations
- DONATOR Name of donator, sometimes also year of donation. 75% are empty; 458 different values including spelling and mark-up variations
- INGEKOMEN Date of acceptance by the museum. 46% are empty
- ETIKET GEGEVENS Serial number on label. 47% are empty; 1796 different values including spelling and mark-up variations; different formats
- GEPRINT Auxiliary field
- BIJZONDERHEDEN Free text fields with comments on anything that does not fit into the other fields. 42% are empty; 2465 values including spelling variations; mostly Dutch.

Database information

- S_GUID Unique identifier automatically generated by Microsoft Access
- REG NUMMER Numerical registry index. 100% filled; values are not always unique
- RECORDER Name of person that entered the record. 100% filled; 7 different values including mark-up variations
- RECORDER DATE Date the record was entered.
- RECORDER TIME Time the record was entered.