Probabilistic Name and Address Cleaning and Standardisation

Peter Christen
Department of Computer Science
Australian National University
Canberra ACT 0200, Australia
Peter.Christen@anu.edu.au

Tim Churches
Centre for Epidemiology and Research
New South Wales Department of Health
Locked Mail Bag 961, North Sydney NSW 2059, Australia
tchur@doh.health.nsw.gov.au

Justin Xi Zhu
Department of Computer Science
Australian National University
Canberra ACT 0200, Australia

ABSTRACT

In the absence of a shared unique key, an ensemble of non-unique personal attributes such as names and addresses is often used to link data from disparate sources. Such data matching is widely used when assembling data warehouses and business mailing lists, and is a foundation of many longitudinal epidemiological and other health related studies.

Unfortunately, names and addresses are often captured in non-standard and varying formats, usually with some degree of spelling and typographical errors. It is therefore important that such data is transformed into a clean and standardised format before it is further processed.

Traditional approaches for cleaning and standardisation of personal information have been based on domain-specific rules that need considerable configuration by highly skilled end users. In this paper we describe an alternative approach based on probabilistic hidden Markov models. Experiments on various health-related administrative data sets show that, compared to a rules-based approach, the probabilistic system is less cumbersome and more flexible to use and, for more complex data, produces more accurate results.

Keywords

Hidden Markov models, data cleaning, data mining, record linkage, biomedical informatics, epidemiology.

1. INTRODUCTION

Most real world data collections contain noisy, incomplete and incorrectly formatted information. Thus data cleaning and standardisation are important first steps in data pre-processing, before such data can be stored in data warehouses or used for further analysis [8; 21]. The cleaning and standardisation of personal names and addresses is especially important for data integration, to make sure that no misleading or redundant information is introduced (e.g. duplicate records). Related to data integration is data linkage, the task of linking together records belonging to the same entity (patient, customer, business) from one or more data sets. Also called record linkage [7], data linkage is important in many domains such as longitudinal epidemiological studies, census related statistics, cleaning of mailing lists, and fraud and crime detection systems.

The main task of data cleaning and standardisation is the conversion of the raw input data into well defined, consistent forms and the resolution of inconsistencies in the way information is represented or encoded. Personal data is often captured and stored with typographical and phonetical variations. Moreover, such data may be recorded or captured in various, possibly obsolete, formats, and data items may be missing or contain errors. For personal data to be useful and valuable, it needs to be cleaned and standardised into a well defined format. For example, nicknames should be expanded into their full names, various abbreviations should be converted into standardised forms, and postcodes should be validated using official postcode lists. In most settings it is desirable to be able to detect and remove duplicate records from a data set, in order to reduce costs for business mailings or to improve the accuracy of a data analysis. De-duplication corresponds to linking a data set with itself.

Where a unique entity identifier or key is shared by all the data sets to be linked or deduplicated, the process of record linkage is trivial. However, this is often not the case. In many data analysis and data mining projects the data required for analysis is contained in two or more separate databases, which do not share a common unique entity identifier. In such cases, probabilistic or other record linkage techniques need to be used to merge the data [5; 25]. Typically a range of non-unique personal data items such as names, addresses and dates (e.g. date of birth) are used to link together records belonging to the same entity (e.g. patient or customer). Data linkage can be used to improve data quality and integrity, to allow reuse of existing data sources for new studies, and to reduce costs and efforts in data acquisition. In order to maximise the likelihood of successful data linkage, data must be cleaned and standardised. For example, comparing the name component from Figure 1 as one string '`Doc Peter Miller`' with a variation of the same name, like '`Mr. Miller, Peter`' would result in a non-match, whereas properly cleaned and standardised into the three name fields title, given name and surname, only the title '`mister`' would differ from the title '`doctor`' thus resulting in a partial match.

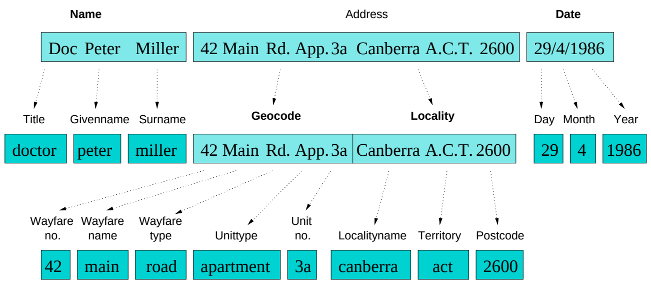

Figure 1: Example name and address standardisation.

Historical collections of administrative health data and transactional business databases nowadays contain many tens or even hundreds of millions of records, with new data being added at the rate of millions of records per annum. Although computing power has increased tremendously in the last few decades, large-scale data cleaning and linking is still a resource-intensive process. There have been relatively few advances over the last decade in the way in which data linkage is undertaken. Only recently [1; 6; 13; 21; 25] researchers started to explore this area, and the first encouraging results on using machine learning techniques are now being reported [2; 4; 15; 23; 24].

The processes of data standardisation and data linkage have various names in different user communities. While statisticians and epidemiologists speak of record or data linkage [5; 7], the same process is often referred to as data scrubbing, pre-processing, or data cleaning [6; 13; 21] by computer scientists and in the database community, whereas it is sometimes called merge/purge processing [9], data integration [3] or ETL (extraction, transformation and loading) in commercial processing of customer databases or business mailing lists. Historically, the statistical and the computer science community have developed their own techniques, and until recently few cross-references could be found.

This paper reports on a project that aims to developing new and improved techniques for data standardisation and data linkage for biomedical data sets. The focus is on improved performance, allowing to standardise and link larger data sets, and on improved quality by using techniques from machine learning and data mining. The prototype software **Febrl**, for *Freely extensible biomedical record linkage*, is currently under development[1]. **Febrl** is written in the free, open-source Python [12] programming language and is itself published under a free, open-source license allowing researchers to customise the software for their particular needs. We hope that **Febrl** will allow biomedical and other researchers to standardise and link their data sets at reduced cost, due to both the free availability of the software and the reduction of human resources needed to use it.

In Section 2 we introduce the task of data cleaning and standardisation of personal names and addresses in more detail. While traditional approaches have been based on rules that need to be customised by the user according to her or his data sets and linkage needs, in Section 3 we present an alternative technique using probabilistic hidden Markov models (HMMs). Instead of having to write sophisticated and complex rules that have to be adjusted for various data sets, the HMM approach needs training records from the data set to be standardised. In Section 3.1 we show how such training data can be created semi-automatically with much less effort and skill than is required to write rules, and in Section 4 we present first results showing that the probabilistic approach can result in highly accurate standardised data by using only small training sets. Finally, Section 5 presents an overview of related work, and Section 6 gives an outlook on current and future directions for this project.

2. CLEANING AND STANDARDISING PERSONAL INFORMATION

The aim of the data cleaning and standardisation process is to transform the raw input data records containing names, addresses and other personal information (like date of birth or gender) into a well defined and consistent form, as shown in Figure 1. Personal information can be categorised into five broad classes: **names**, **addresses**, **dates** (such as date-of-birth), **categorical attributes** (such as sex or country-of-birth) and **identifying numbers** (such as tax file or Medicare numbers). This paper deals with the first two of these classes.

Addresses can be further separated into two parts, namely the **geocode** part which contains street and postal address information, and the **locality** part which contains town or suburb names, postcode, as well as state and country information. As can be seen from Table 1, each component is further split into several output fields each containing a basic piece of information.

Name	Geocode	Locality
title	wayfare_number	postcode
gender_guess	wayfare_name	locality_name
givenname	wayfare_type	locality_qualifier
alt_givenname	wayfare_qualifier	territory
surname	unit_number	country
alt_surname	unit_type	
	property_name	
	institution_name	**Date**
	institution_type	day
	postaddress_number	month
	postaddress_type	year

Table 1: Supported output fields.

We assume that the raw input data records are stored as text files or database tables and each of the input components is made of one or more text strings. The task is then to allocate the words and numbers from the raw input into the appropriate output fields, and to clean and standardise the values in these output fields.

Our approach to data cleaning and standardisation is based on the following three steps. Firstly, the input strings are *cleaned*. Secondly, they are split into a list of words, numbers and characters, which are then *tagged* using look-up tables (mainly for names and addresses) and some hard-coded rules (e.g. for numbers, hyphens or commas). Finally these tagged lists are *segmented* into output fields using probabilistic hidden Markov models. We discuss each step in more detail in the following three sections.

[1] See project web site: `http://datamining.anu.edu.au/linkage.html`

2.1 Cleaning

The cleaning step involves converting all letters into lower case followed by various general corrections of sub-strings using correction lists, which are stored in text files and can be modified by the user. Such correction lists are made of pairs of strings `original:replacement`. If an `original` string is found in the raw input sting, it is replaced by the corresponding `replacement` string. For example, variations of *'known as'*, such as `'a.k.a.'` or `'aka'` are all replaced with the string `'known as'`. Various kinds of brackets and quoting characters are all replaced with a vertical bar `'|'`, which facilitates tagging and segmenting in the subsequent steps. Correction lists also allow the replacement string to be an empty string, in which case an original string found in the input is removed. For example, the original string `'abbrev'` can be removed from an input record, by setting is replacement string to `''`. Note that the **name** component has a different correction list than the **geocode** and **locality** components, because some corrections are specific to names or addresses. The output of the cleaning step is a cleaned string ready to be tagged in the next step.

2.2 Tagging

After an input component string has been cleaned, the next step is to split it at white-space boundaries into a list of words, numbers, characters, punctuation marks and other possible separators. Using look-up tables and some hard-coded rules each of the list elements is assigned one or more *tags*. The hard-coded rules include, for example, tagging an element as a hyphen, a comma, a slash, a number or an alphanumeric word, while most of the other tags (titles, given names, surnames, postcode, locality names, wayfare and unit types, countries, etc.) are assigned to words if they are listed in one of the look-up tables. A list of all supported tags is given in Table 2. These look-up tables are loaded from text files, which can be modified by the user. If a word (or a word sequence) is found in a look-up table, it is not only tagged, but it is also replaced by its corresponding corrected entry in the look-up table. It is possible that a word is listed in more than one look-up table. Consequently, it will be assigned more than one tag (see for example the name word `'peter'` below). Words which are not found in any look-up table and which do not match any of the hard-coded tagging rules are assigned the `'UN'` (unknown) tag. Title words like `'doctor'`, `'doc'`, `'md'` and `'phd'` for example will all be assigned the title word tag `'TI'`, and they will be replaced with the standardised word `'dr'`.
The look-up tables are searched using a *greedy* matching algorithm, which searches for the longest tuple of elements that match an entry in the look-up tables. For example, the tuple of words `('macquarie','fields')` will be matched with an entry in a look-up table with the locality name `'macquarie fields'`, rather than with the shorter entry `'macquarie'` from the same look-up table.
The output of the tagging step is a list of elements (words, numbers and separators) and a corresponding list of tags.

Example: Assume the raw input string of the name component is `'Doc. peter Paul MILLER'`, which is converted by the cleaning step into the string `'doc peter paul miller'`. Assuming that `'doc'` is listed in the title look-up table (and corrected into `'dr'`), it will be assigned a `'TI'` tag. Further assuming `'peter'` is listed in both the male given name and the surname look-up tables, it is assigned the two tags `'GM'` and `'SN'`. If the next word `'paul'` is only found in the male given name look-up table, it will only be assigned a `'GM'`. Finally, assume the name `'miller'` is only found in the surname look-up table, so it will be assigned the tag `'SN'`. Because `'peter'` has been assigned two tags, the following two permutations of tag sequences are possible:

```
['dr', 'peter', 'paul', 'miller']
['TI',   'GM',   'GM',    'SN'  ]
['TI',   'SN',   'GM',    'SN'  ]
```

The question is now which of these two tag sequences is the most likely one, and how should the elements of the word list be assigned to the appropriate output fields? This problem is solved using probabilistic hidden Markov models in the segmentation step as discussed below.

Tag	Description	Name	Geocode/ Locality
`LQ`	Locality qualifier words	–	LT
`LN`	Locality (town) names	–	LT
`TR`	Territory (state) names	–	LT
`CR`	Country names	–	LT
`IT`	Institution type words	–	LT
`IN`	Institution names	–	LT
`PA`	Postal address type words	–	LT
`PC`	Postcodes	–	LT
`UT`	Unit type words	–	LT
`WN`	Wayfare names	–	LT
`WT`	Wayfare type words	–	LT
`ST`	Saint names	LT	LT
`TI`	Title words	LT	–
`SN`	Surnames	LT	–
`GF`	Female given names	LT	–
`GM`	Male given names	LT	–
`PR`	Name prefix words	LT	–
`SP`	Name separators (like *'known as'*)	LT	–
`BO`	*'baby of'* and similar sequences	LT	–
`NE`	The word *nee* (surname or *'born'*)	LT	–
`II`	One-letter words (initials)	–	HC
`HY`	Hyphen *'-'*	HC	HC
`CO`	Comma *','*	HC	HC
`SL`	Slash *'/'*	HC	HC
`N4`	Numbers with four digits	–	HC
`NU`	Other numbers (not four digits)	HC	HC
`AN`	Alpha-numeric words	HC	HC
`VB`	Vertical bar *'|'* (various brackets)	HC	HC
`RU`	Rubbish	LT	LT
`UN`	Unknown	LT	LT

Table 2: Supported tags, based either on look-up tables (LT) or hard coded rules (HC).

2.3 Segmenting

Having a list of elements (words, numbers, characters and separators) and one or more corresponding tag lists, the task

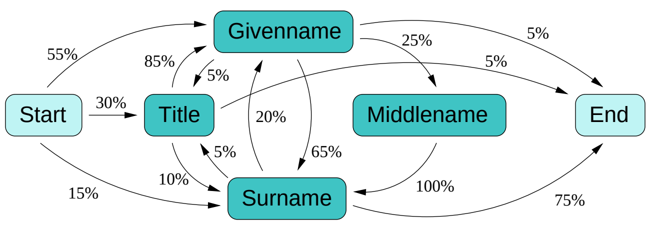

Figure 2: Simple example hidden Markov model for name component.

From state	*Start*	Title	Given name	Middle name	Sur- name	*End*
	To state					
Start	–	0.30	0.55	0.0	0.15	–
Title	–	0.0	0.85	0.0	0.10	0.05
Given- name	–	0.05	0.0	0.25	0.65	0.05
Middle- name	–	0.0	0.0	0.0	1.00	0.0
Surname	–	0.05	0.20	0.0	0.0	0.75
End	–	–	–	–	–	–

Obser. symbol	*Start*	Title	Given name	Middle name	Sur- name	*End*
	State					
TI	–	0.96	0.01	0.01	0.01	–
GM	–	0.01	0.35	0.33	0.15	–
GF	–	0.01	0.35	0.27	0.14	–
SN	–	0.01	0.09	0.14	0.45	–
UN	–	0.01	0.20	0.25	0.25	–

Table 3: Example name HMM transition and observation probabilities.

is now to assign these elements to the appropriate output fields. Traditional approaches have used rules (such as *'if an element has a tag 'TI' then the corresponding word is assigned to the* **'title'** *output field.'*). Instead, we use probabilistic hidden Markov models [20]. The advantages of hidden Markov models are their robustness with respect to previously unencountered input sequences, as well as the fact that they can be trained by clerical staff, rather than requiring high level analysis and programming skills to create complex rules that accommodate all special cases for a given data set.

3. DATA SEGMENTATION USING HIDDEN MARKOV MODELS

Traditional data cleaning and standardisation systems apply various rule-based approaches to the task of parsing raw data. For example, **AutoStan** [14] uses an initial lexicon-based tokenisation phase followed by a re-entrant rule-based parsing and token re-writing phase. The approach presented in this paper also uses lexicon-based tokenisation (or tagging, as described above), but then uses a probabilistic approach based on hidden Markov models to assign each element in the cleaned and tagged input list to a particular output field.

Hidden Markov models [20] (HMMs) were developed in the 1960s and 1970s and are widely used in speech and natural language processing. They are a powerful machine learning technique, able to handle new forms of data in a robust fashion. They are computationally efficient to develop and evaluate. Only recently have HMMs been used for name and address standardisation [2; 22].

A HMM is a probabilistic finite state machine made of a set of states, transition edges between these states and a finite dictionary of discrete observation (output) symbols. Each edge is associated with a transition probability, and each state emits observation symbols from the dictionary with a certain probability distribution. Two special states are the *start* and *end* state. Beginning from the start state, a HMM generates a sequence of observation symbols $O = o_1, o_2, \ldots, o_k$ by making $k-1$ transitions from one state to another until the end state is reached. Observation symbol $o_i, 1 \leq i \leq k$ is generated in state i based on this state's probability distribution of the observation symbols. The same output sequence can be generated by various paths through a HMM with different probabilities. Given an observation sequence, one is often interested in the most probable path through a given HMM that generated this sequence. The *Viterbi* [20] algorithm is an efficient way to compute this most probable path for a given observation sequence.

The distribution of both transition and observation probabilities are learned using training data. Each training record is an example path and observation sequence. While the dictionary of output symbols can be created using the training data, the states of a HMM are normally fixed and have to be defined before training. Because training data often does not cover all possible combinations of states and observations, during testing and application of a HMM unseen and unknown data is encountered. To be able to deal with such cases, *smoothing* techniques [2] need to be applied which enable unseen data to be handled more efficiently. These techniques – such as *Laplace* or *absolute discounting* – basically assign small observation probabilities to all unseen observations symbols in all states. For states that have more distinct words during training, they are also expected to encounter unknown symbols more frequently. Smoothing techniques reflect this fact by assigning unseen symbols in those states a higher relative probability [2].

Figure 2 shows a simple HMM example for the name component with six states. The *start* and *end* states are both virtual states that are not actually stored in a HMM, as no symbols are emitted in these states. Instead of the *start* state a list of initial state probabilities is used (i.e. probabilities that give the likelihood of a sequence starting in a certain state). In the given example, a name starts with a 55% likelihood with a *Givenname* and is followed with a (conditional) probability of 65% by a *Surname*, or 25% probability with a *Middlename*, and so on.

Instead of using the original words, numbers and other elements from the input records directly, the tag sequences (as discussed in Section 2.2) are used as HMM observation symbols in order to make the derived HMMs more general, i.e. to allow HMMs to be trained on one data set and then be used with other similar, but distinct data sets, with little or no loss of performance. Using tags also limits the size of the observation dictionary. Once a HMM is trained, sequences of tags (one tag per input element) as generated in the tagging step can be given as input to the *Viterbi* algorithm, which returns the most likely path (i.e. state sequence) of the given tag sequence through the HMM, plus the corresponding probability. The path with the highest probability is then taken and the corresponding state sequence will be used to assign the elements of the input to the appropriate output fields.

Example: Assuming the same name example as above, the input name string `'Doc. peter Paul MILLER'` is cleaned and tagged as explained in Section 2.2 into the following word list and tag sequences:

```
['dr', 'peter', 'paul', 'miller']
['TI',   'GM',    'GM',    'SN'  ]
['TI',   'SN',    'GM',    'SN'  ]
```

These two tag sequences are given to the *Viterbi* algorithm and using the name HMM from Figure 2 with transition and observation probabilities as listed in Table 3, the first tag sequence `['TI','GM','GM','SN']` is assigned to the following path through the HMM (with the corresponding observation symbols in brackets):

```
Start -> Title (TI) -> Givenname (GM) ->
 Middlename (GM) -> Surname (SN) -> End
```

The resulting probability of this path is:

```
0.30 * 0.96 * 0.85 * 0.35 * 0.25 * 0.33 * 1.00 *
           0.45 * 0.75 = 0.0023856525
```

with `0.30` being the transition probability from state `Start` to state `Title`, then `0.96` being the probability that the symbol `'TI'` is observed in state `Title`, `0.85` being the transition probability from the `Title` to the `Givenname` state, and so on. The most probable path for the second sequence `['TI','SN','GM','SN']` is:

```
Start -> Title (TI) -> Givenname (SN) ->
 Middlename (GM) -> Surname (SN) -> End
```

which would result in a probability of:

```
0.30 * 0.96 * 0.85 * 0.09 * 0.25 * 0.33 * 1.00 *
           0.45 * 0.75 = 0.0006134534
```

Thus the first tag sequence is the most likely one, as expected. Using the HMM states, the elements of the input word list are then associated with the corresponding output fields. In this example `'dr'` will become the title, `'peter'` will become the given name, `'paul'` the middle name (or alternative given name) and `'miller'` the surname.

3.1 Hidden Markov Model Training

In our data cleaning and standardisation system we are using one HMM for names and one for addresses (geocode and locality) with the corresponding states as listed in Table 4 and Table 5. Many of the output fields can contain more than one word, therefore the HMMs appropriately have two states for these elements which are then both assigned to the corresponding output field.
Both the transition and observation probabilities for the HMMs have to be trained using collections of records that have been annotated manually, and which are taken from the same (or a similar) data set which will be used for data standardisation. Using these *training records* a HMM *learns* the characteristics of a data set.

State	Description
`titl`	Title state
`baby`	State for *baby of*, *son of* or *daughter of*
`knwn`	State for *known as*
`andor`	State for *and* or *or*
`gname1`	Given name state 1
`gname2`	Given name state 2
`ghyph`	Given name hyphen state
`gopbr`	Given name opening bracket state
`gclbr`	Given name closing bracket state
`agname1`	Alternative given name state 1
`agname2`	Alternative given name state 2
`coma`	State for comma
`sname1`	Surname state 1
`sname2`	Surname state 2
`shyph`	Surname hyphen state
`sopbr`	Surname opening bracket state
`sclbr`	Surname closing bracket state
`asname1`	Alternative surname state 1
`asname2`	Alternative surname state 2
`pref1`	Name prefix state 1
`pref2`	Name prefix state 2
`rubb`	Rubbish state, for elements to be thrown away

Table 4: Name hidden Markov model states.

Thus, instead of requiring highly trained programming staff to maintain a large number of rules that cover all kinds of special cases and exceptions, and that have to be modified for each new given data set, the data to train a HMM can be created by clerical staff within a couple of days for a new data source. Furthermore, this training process can be accelerated by *bootstrapping* it with training data sets derived from other, similar data sources.
The training data consists of sequences with one or more `tag:hmm_state` pairs. Each sequence is a training record that is given to the HMM, and the HMM learns the characteristics of a data set by using all training examples that it is given during training. Maximum likelihood estimates (MLEs) for the matrix of transition and observation probabilities for a HMM are derived by accumulating frequency counts of each type of transition and output symbol (tag) from the training records. Because frequency-based MLEs are used, it is important that the records in the training data set(s) are reasonably representative of the overall data set(s) to be standardised. However, HMMs are quite robust to unseen data and are not overly troubled if the records in the training data set(s) do not represent an unbiased sample of records from the target data. For example, it is possible to add training records which represent unusual records without degrading the performance of the HMMs on more typical records. HMMs also degrade gracefully, in that they still perform well even with records with a previously unencountered structure. A simple set of training examples for a name component HMM might look like:

```
GF:gname1, SN:sname1
UN:gname1, SN:sname1
GF:gname1, GM:gname2, UN:sname1
GF:gname1, GM:sname1
GF:gname1, UN:gname2, SN:sname1
```

Each line in the example above corresponds to one training record, and contains a sequence that corresponds to a partic-

State	Description
wfnu	Wayfare number state
wfna1	Wayfare name state 1
wfna2	Wayfare name state 2
wfql	Wayfare qualifier state
wfty	Wayfare type state
unnu	Unit number state
unty	Unit type state
prna1	Property name state 1
prna2	Property name state 2
inna1	Institution name state 1
inna2	Institution name state 2
inty	Institution type state
panu	Postal address number state
paty	Postal address type state
hyph	State for hyphen
sla	State for slash
coma	State for comma
opbr	Opening bracket state
clbr	Closing bracket state
loc1	Locality name state 1
loc2	Locality name state 2
locql	Locality qualifier state
pc	Postcode state
ter1	Territory name state 1
ter2	Territory name state 2
cntr1	Country name state 1
cntr2	Country name state 2
rubb	Rubbish state, for elements to be thrown away

Table 5: Address hidden Markov model states.

ular path through the various (hidden, unobserved) states of the HMM together with the corresponding observation symbols (tags).
For a new data set HMMs for names and addresses can be created in four main steps. First, a small number (around 100) of records from the original data set are selected randomly and tagged as described in Section 2.2. The output of this step is one or more tag sequences for each input record (name or address component). These training records then have to be edited manually by the user by choosing the correct tag sequence (if more than one has been created for one input record) and by adding the appropriate HMM state to each of the tags in the selected tag sequence. The resulting small number of training records (like the examples above) are then used in the second step to train a first rough HMM, which in the third step is used to create a larger set of training records, again randomly selected from the original data set. These training records now already have a tag sequence that includes HMM states. The user then has to go through these training records and correct tags or HMM states if they are wrong. In the file containing the training records, the original input is commented out, so the user can easily check if a training record is correct or needs to be modified. The two examples of address training records below show the original input string, the cleaned input string at the end of the tagging routine (the tagged word list concatenated back into a string) and the corresponding tag sequence which will be taken as training example. Note that both the original input and the tagged input are commented using the Python comment character hash.

```
# '2 richard st lewisham 2049 nsw'
# '2 richard street lewisham 2049 new_south_wales'
NU:wfnu, UN:wfna1, WT:wfty, LN:loc1, PC:pc, TR:ter1

# '42/131 miller pl manley 2095 nsw'
# '42 / 131 miller place manly 2095 new_south_wales'
NU:unnu, SL:sla, NU:wfnu, UN:wfna1, WT:wfty,
LN:loc1, PC:pc, TR:ter1
```

Using this *bootstrapping* approach a single user can create and validate in one or two days work enough training records to be able to standardise real world data sets with a high accuracy as shown in the following section.

4. STANDARDISATION RESULTS

We developed and evaluated the **Febrl** system using three routinely-collected health data sets which contain personal identifying details. Access to these data sets for the purpose of this project was approved by the Australian National University Human Research Ethics Committee and by the relevant data custodians within the NSW Department of Health. The data sets used in this project were held on secure computing facilities at the Australian National University and the NSW Department of Health head offices. Access to the data sets used in this project was strictly limited to the investigators. All investigators, as well as the system administrators of the computing facilities, were required to sign a confidentiality agreement which apprised them of their responsibilities as well as the legislative protection (and associated criminal penalties for misuse) afforded to the data.
In order to minimise the invasion of privacy which is necessarily associated with every use of identified data, all medical and health status details and other personal details apart from name (on two of the data sets), date-of-birth (on one of the data sets), sex and residential address were removed from the data sets used in this project. The **Febrl** software under development is multi-platform capable, and we were able to tag, clean and segment names and address on both 64-bit Unix and 32-bit Windows platforms without problems or modifications.

4.1 Address Standardisation

The performance of the **Febrl** system with typical Australian address data was evaluated using two data sets. The first was a subset of approximately 1 million addresses taken from uncorrected electronic copies of NSW death certificates as completed by medical practitioners and coroners in the years 1988 to 2002. The information systems which captured these data underwent a number of major changes during this period. The majority of these data were entered from handwritten forms.
The second data set was a random sample of 1,000 records of residential addresses drawn from the NSW Inpatient Statistics Collection for the years 1993 to 2001. This collection contains abstracts for every admission to a public- or private-sector acute care hospital in NSW. Most of the data are extracted from a range of computerised hospital information systems, with a smaller proportion entered from paper forms.

A number of tests were carried out:

1. An initial *bootstrap* hidden Markov model (HMM) was trained using 100 random death certificate (DC) records, and this was used to form a larger training set of 1,100 randomly chosen DC records. A HMM derived from this second training set was then used to standardise 50,000 randomly chosen DC records, and records with unusual patterns of observation symbols (with a frequency of six or less) were examined, corrected and added to the training set if the results produced by the HMM were incorrect. A new HMM was then derived from this augmented training set and the process repeated a further three times, resulting in the addition of approximately 250 extra training records (bringing the total number of training records to 1,450). The HMM which emerged from this process, designated HMM1, was used to standardise 1,000 randomly chosen DC test records and the accuracy of the standardisation was assessed. Laplace smoothing is used in this and all subsequent tests.

2. HMM1 was then used to standardise 1,000 randomly chosen Inpatient Statistics Collection (ISC) records, and the accuracy was assessed. In other words, a HMM trained using one data source (DC) was used to standardise addresses from a different data source (ISC) without any retraining of the HMM.

3. An additional 1,000 randomly chosen address training records derived from the Midwives Data Collection (MDC) were then added to the 1,450 training records described above, and this larger training set was used to train HMM2. HMM2 was then used to re-standardise the same sets of randomly chosen test records described in steps 1 and 2 above, and the results were evaluated.

4. Approximately 60 training records, based on archetypes of those records which were wrongly standardised in all of the preceding tests, were then added to the training set to produce HMM3. HMM3 was then used to re-standardise the same DC and ISC test sets. Thus, HMM3 could be considered an "overfitted" model for the particular records in the two test sets, although in practice researchers are likely to use such "overfitting" to maximise standardisation accuracy for the specific data sets used in their particular study.

5. Finally, by way of comparison, the same two 1,000 record test data sets were standardised using the commercial **AutoStan** [14] software in conjunction with a rule set which had been developed and refined by two of the investigators (TC and KL) over the course of several years for use with ISC (but not DC) address data.

For all tests, records were judged to be accurately standardised when all of the elements present in the input address string, with the exception of punctuation, were allocated to the correct output field, and the values in each output fields were correctly transformed to their canonical form where required. Thus, a record was judged to have been incorrectly standardised if any element of the input string was not allocated to an output field, or if any element was allocated to the wrong output field. Due to resource constraints, the investigators were not blinded to the process used to standardise the records. Results are shown in Table 6.

	HMM/Method			
Test Data Set (1,000 records each)	HMM 1	HMM 2	HMM 3	Auto Stan
Death Certificates	95.7	96.8	97.6	91.5
Inpatient Statistics Collection	95.7	95.9	97.4	95.3

Table 6: Proportion of correctly standardised address records (all numbers are percentages).

These results indicate that the HMM approach described in this paper produces standardisation accuracies which are comparable to those produced by a well-established rule-based system when used on the data set for which those rules were developed, and superior results when used on a different data set. In other words, the HMM approach appears to be less sensitive to the particular characteristics of the data source for which it was developed than a widely-used rule-based system.

In addition, the results indicate that although the HMMs are trained using maximum likelihood estimates, they are quite robust with respect to the source of the training data and their performance can even be improved by the addition of a small number of unrepresentative training records which represent "difficult" cases.

In those records which were not accurately standardised by the HMMs, an average of 83 per cent of all data elements present in the input record were allocated to the correct output fields – in other words, even these incorrectly standardised records would have considerable utility. In only two test records (out of 2,000) were all of the elements wrongly assigned, and both of these were foreign addresses in non-English speaking countries. The performance of **AutoStan** in this respect was similar.

Finally, the addition of a small number of deterministic post-processing rules is expected to yield even higher accuracies in future versions of the **Febrl** system.

4.2 Name Standardisation

Accuracy measurements on names were conducted using a subset of the *NSW Midwives Data Collection (MDC)*. This subset contained 962,776 records with personal information (but no medical details) of women who had given birth in New South Wales, Australia, over a ten year period (1990-2000). Most of the data was entered from hand-written forms, although some of the data for the latter years was extracted directly from a variety of computerised obstetric information systems.

A random subset of 10,000 records (around a 1% sample) with a non-empty name component were selected and split into 10 test sets each containing 1,000 records. A 10-fold cross validation study was performed, with each of the folds having a training set of 9,000 records and the remaining 1,000 records being the test set. The training records were tagged in about 10 person-hours using the bootstrapping method as explained in Section 3.1. Hidden Markov models were then trained without smoothing, and both with Laplace and absolute discount [2] smoothing, respectively.

Compared to the variation in the format of residential addresses, names in the MDC are rather homogeneous. Out of the 10,000 randomly selected names, around 85% were of the simple form *'givenname surname'*, and further 9% of either the form *'givenname givenname surname'* or *'givenname surname surname'*. Thus the trained HMMs had very few non-zero transition probabilities, and many HMM states were not linked (with non-zero transitions) to the *active* HMM part.

Table 7 shows accuracy results of the HMM and a rules-based name standardisation algorithm which is also implemented in the **Febrl** prototype software. Given the simple form of most names, the rules based approach was very accurate, achieving 97% and better. The variations in the HMM approach were much higher, with up to 17% of names wrongly standardised. There were almost no differences in the effect of the smoothing method, therefore we only report results based on the unsmoothed HMM here. In many cases the same errors in standardisation occurred for both unsmoothed and smoothed HMMs, with a maximum of less than five records standardised differently per 1,000 records.

	Min	Max	Average	StdDev
HMM	83.1	97.0	92.0	±4.7
Rules	97.1	99.7	98.2	±0.7

Table 7: Name standardisation with 10-fold cross-validation (all numbers are percentages).

Especially problematic seemed to be names with either two given names or two surnames. Often the HMMs misclassified the middle name as first surname instead of second given name. This is due to the large number of names with the simple form *'givenname surname'* which results in a very high transition probability from the first given name state `'gname1'` to the first surname state `'sname1'`. A second given name therefore is often assigned as a first surname, and the real surname as a second surname. Other problems were surnames that were listed in a given name look-up table only and thus tagged with a given name tag (`'GF'` or `'GM'`), in which case the HMMs wrongly assigned the second name to the given name instead of the surname output field.

It is clear from these results that some additional rules could usefully be added to the rules-based name standardisation, as well as some rule-based post-processing to the HMM segmentation process. One simple example (both for the rules and HMM approaches) is: *'if no surname is given, but two given name are present, re-assign the second given name word to surname'* (assuming given names are generally written before surnames, otherwise the opposite way).

5. RELATED WORK

The terms *data cleaning* (or *data cleansing*), *data standardisation*, *data scrubbing*, *data pre-processing* and *ETL* (extraction, transformation and loading) are used synonymously to refer to the general tasks of transforming source data (often derived from operational, transactional information systems) into clean and consistent sets of records which are suitable for loading into databases and data warehouses, and for linking with other data sets [21]. Once the data has been standardised, the central task of data linkage is to identify records in the source data sets that represent the same real-world entity. In the computer science literature, this process is also called the object identity or merge/purge problem [9]. Fuzzy techniques and methods from information retrieval have been used to address the data linkage problem, with varying degrees of success. One approach is to represent text (or records) as document vectors and compute the *cosine distance* [3] between such vectors. Another possibility is to use an *SQL* like language [6] that allows approximate joins and cluster building of similar records, as well as decision functions that decide if two records represent the same entity. Other methods [13] include statistical outlier identification, pattern matching, clustering and association rules based approaches. Sorting data sets (to group similar records together) and comparing records within a sliding window [9] is a technique similar to blocking as applied by traditional record linkage approaches. The accuracy of the linkage can be improved by having smaller window sizes and performing several passes over the data using different (often compound) keys, rather than having a large window size but only one pass. This corresponds to applying several blocking strategies in a record linkage process.

Machine learning techniques have been applied to data linkage in recent years. The authors of [4; 24] describe a hybrid system that in a first step uses unsupervised clustering on a small sample data set to create data that can be used by a classifier in the second step to classify records into links and non-links. The authors do not directly address the problem of data cleaning and standardisation, rather they use approximate string comparison algorithms to be able to deal with variations in strings. Clustering of large and high-dimensional data sets with applications to matching is discussed in [15]. The authors propose a two step clustering algorithm that in the first step uses cheap approximate distance metrics to form overlapping canopies, which are in a second step clustered using traditional approaches. Another approach [16] learns field specific string-edit distance weights and a binary classifier based on support vector machines (SVM) to find duplicate records in text databases.

Commercial software for data cleaning and standardisation is available from various vendors (see the project web page for a non-exhaustive list of data cleaning and data integration software). Most of these products use proprietary technologies, but many are rules based. The **AutoStan / AutoMatch** [14] suite of data cleaning and linkage software for example improved on simple (or far-from-simple) regular expression rules by using an initial lexicon-based tokenisation phase followed by a re-entrant rule-based parsing and token re-writing phase. Probabilistic approaches as the one presented in this paper on the other hand allow the creation of training data in a much less time consuming fashion.

While hidden Markov models have traditionally been applied in areas like signal and speech processing [20], text recognition, and image processing, only recently they have been applied to the tasks of information extraction [22] (e.g. extracting names, titles and keywords from publication abstracts) and segmentation of addresses and bibliographic records [2], where an input string containing an address has to be segmented into well defined output fields. The approach discussed in this paper differs in that instead of using address words directly, only tags are given to the HMMs, which results in a system that can handle unseen data more robustly and is also computationally more efficient due to its smaller dictionaries of output symbols.

6. CONCLUSIONS AND OUTLOOK

In this paper we presented a probabilistic approach for the task of cleaning and standardising personal names and addresses using hidden Markov models. This process is not only an important first step before data can be loaded into databases or data warehouses, it is also necessary before data can be linked or integrated with other data. We have shown that the probabilistic approach is not only easier and less cumbersome to use compared to the traditional rule based approach, it can also result in a higher accuracy. The methods presented are part of a prototype software system, which is published under an open source license and which can be downloaded from the project web page (see `http://datamining.anu.edu.au/linkage.html`).

Currently we are developing new methods for data linkage based on the probabilistic approach as developed by Fellegi and Sunter [5] and extended by others. The two main foci are to improve the linkage performance by applying techniques from high-performance and parallel computing, as well as improving the linkage quality by exploring machine learning techniques like clustering and predictive modelling. The aim is to provide researchers and users in the biomedical and related areas with the ability to clean, standardise and link much larger data sets at reduced costs, due to the reduction in human resources needed and the free availability of the software.

Acknowledgments

This project is equally funded by the *Australian National University (ANU)* and the *NSW Department of Health* under an *AICS (ANU-Industry Collaboration Scheme) AICS #1-2001*. The authors would like to thank everybody who supported this project and helped to make it happen: Ole Nielsen, Markus Hegland, Stephen Roberts and David Bulbeck. We are specially grateful to Kim Lim (Centre for Epidemiology and Research, NSW Department of Health) for her input into the design of the prototype software and in helping to test and debug it.

7. REFERENCES

[1] G.B. Bell and A. Sethi, *Matching Records in a National Medical Patient Index*, Communications of the ACM, Vol. 44 No. 9, September 2001.

[2] V. Borkar, K. Deshmukh and S. Sarawagi, *Automatic segmentation of text into structured records*, in Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, California, 2001.

[3] W.W. Cohen, *The WHIRL Approach to Integration: An Overview*, in Proceedings of the AAAI-98 Workshop on AI and Information Integration. AAAI Press, 1998.

[4] M.G. Elfeky, V.S. Verykios and A.K. Elmagarmid, *TAILOR: A Record Linkage Toolbox*, Proceedings of the ICDE' 2002, San Jose, USA, 2002.

[5] I. Fellegi and A. Sunter, *A theory for record linkage.* In Journal of the American Statistical Society, 1969.

[6] H. Galhardas, D. Florescu, D. Shasha and E. Simon, *An Extensible Framework for Data Cleaning*, Technical Report 3742, INRIA, 1999.

[7] L. Gill, *Methods for Automatic Record Matching and Linking and their use in National Statistics*, National Statistics Methodology Series No. 25, London 2001.

[8] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2000.

[9] M.A. Hernandez and S.J. Stolfo, *The Merge/Purge Problem for Large Databases*, in Proceedings of the SIGMOD Conference, San Jose, 1995.

[10] C.W. Kelman, *Monitoring Health Care Using National Administrative Data Collections*, PhD thesis, Australian National University, Canberra, May 2000.

[11] A.J. Lait, and B. Randell, *An Assessment of Name Matching Algorithms*, Technical Report, Department of Computing Science, University of Newcastle upon Tyne, UK 1993.

[12] M. Lutz, *Python Pocket Reference, Second Edition*, O'Reilly and Associates, January 2002.

[13] J.I. Maletic and A. Marcus, *Data Cleansing: Beyond Integrity Analysis*, in Proceedings of the Conference on Information Quality (IQ2000), Boston, October 2000.

[14] *AutoStan and AutoMatch, User's Manuals*, MatchWare Technologies, Kennebunk, Maine, 1998.

[15] A. McCallum, K. Nigam and L.H. Ungar, *Efficient clustering of high-dimensional data sets with application to reference matching*, Knowledge Discovery and Data Mining, 169-178, 2000.

[16] U.Y. Nahm, M. Bilenko and R.J. Mooney, *Two Approaches to Handling Noisy Variation in Text Mining*, in Proceedings of the ICML-2002 Workshop on Text Learning (TextML'2002), pp.18-27, Sydney, Australia, July 2002.

[17] H.B. Newcombe and J.M. Kennedy, *Record Linkage: Making Maximum Use of the Discriminating Power of Identifying Information*, Communications of the ACM, Vol. 5 No. 11, 1962.

[18] L. Philips, *The Double-Metaphone Search Algorithm*, C/C++ User's Journal, Vol. 18 No. 6, June 2000.

[19] E.H. Porter and W.E. Winkler, *Approximate String Comparison and its Effect on an Advanced Record Linkage System*, Research Report RR97/02, US Bureau of the Census, 1997.

[20] L.R. Rabiner, *A tutorial on Hidden Markov Models and selected applications in speech recognition*, in Proceedings of the IEEE, vol. 77, no. 2, February 1989.

[21] E. Rahm and H.H. Do, *Data Cleaning: Problems and Current Approaches*, IEEE Bulletin of the Technical Committee on Data Engineering, Vol. 23 No. 4, December 2000.

[22] K. Seymore. A. McCallum and R. Rosenfeld, *Learning Hidden Markov Model Structure for Information Extraction*, in Proceedings of AAAI-99 Workshop on Machine Learning for Information Extraction, 1999.

[23] V.S. Verykios, A.K. Elmagarmid and E.N. Houstis, *Automating the Approximate Record-Matching Process*, Information Sciences, Vol. 126, July 2000.

[24] V.S. Verykios, A.K. Elmagarmid, M.G. Elfeky, M. Cochinwala and S. Dalal, *On the Completeness and Accuracy of the Record Matching Process*, in Proceedings of the MIT Conference on Information Quality, Boston, MA, October 2000.

[25] W.E. Winkler, *The State of Record Linkage and Current Research Problems*, U.S. Census Bureach Research Report RR99/04, 1999.

[26] W.E. Winkler, *Quality of Very Large Databases*, Research Report RR2001/04, US Bureau of the Census, 2001.

[27] W.E. Yancey, *Frequency-Dependent Probability Measures for Record Linkage*, Research Report RR00/07, Statistical Research Division, US Bureau of the Census, July 2000.

[28] W.E. Yancey, *BigMatch: A Program for Extracting Probable Matches from a Large File for Record Linkage*, Research Report RR 2000-01, Statistical Research Division, US Bureau of the Census, March 2002.