# Automated Probabilistic Address Standardisation and Verification

Peter Christen and Daniel Belacic

Data Mining Group, Australian National University

Contact: **peter.christen@anu.edu.au**

Project web page: **http://datamining.anu.edu.au/linkage.html**

Funded by the ANU, the NSW Department of Health,

and the Australian Research Council (ARC) (LP #0453463)

THE AUSTRALIAN NATIONAL UNIVERSITY

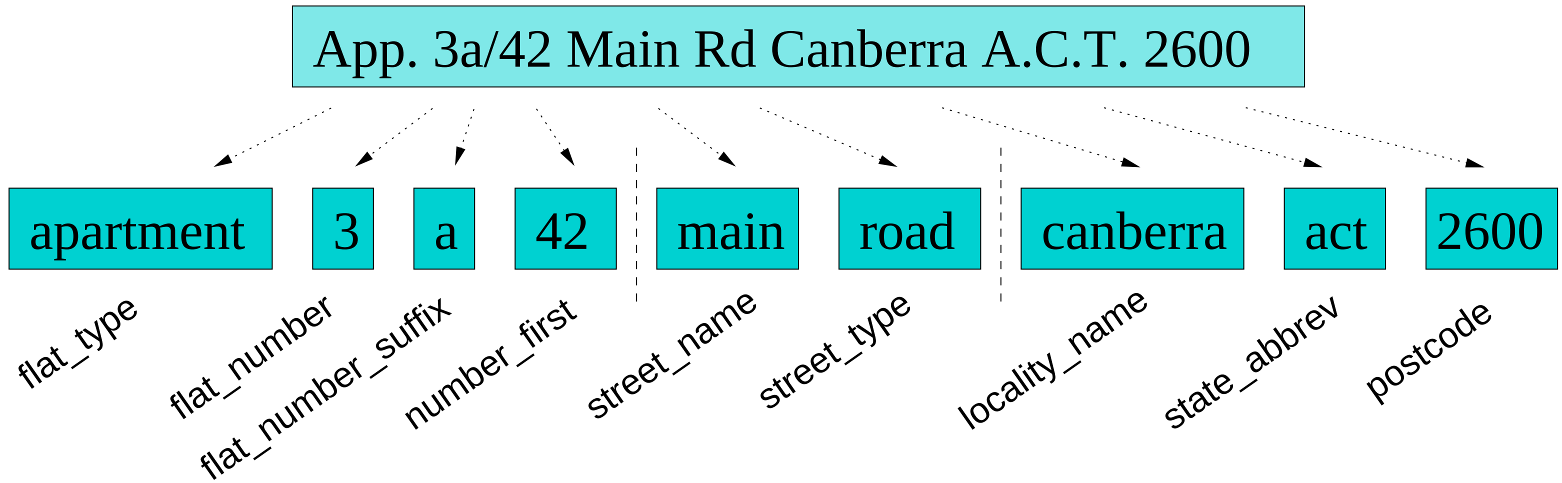
# *Outline*

- Address cleaning and standardisation
- Probabilistic standardisation approaches
- Hidden Markov models for data segmentation
- Our probabilistic approach
- Automated Hidden Markov model training
- Experimental results
- Conclusions and outlook

# *Why address standardisation?*

- Real world data is often *dirty*
  - Typographical and other errors
  - Different coding schemes
  - Missing values
  - Data changing over time
- Addresses (and names) are especially prone to data entry errors
  - Scanned, hand-written, over telephone, hand-typed
  - Same person often provides her/his details differently
  - Different correct spelling variations for proper names (e.g. *Gail* and *Gayle*, or *Dixon* and *Dickson*)

# *Address standardisation tasks*



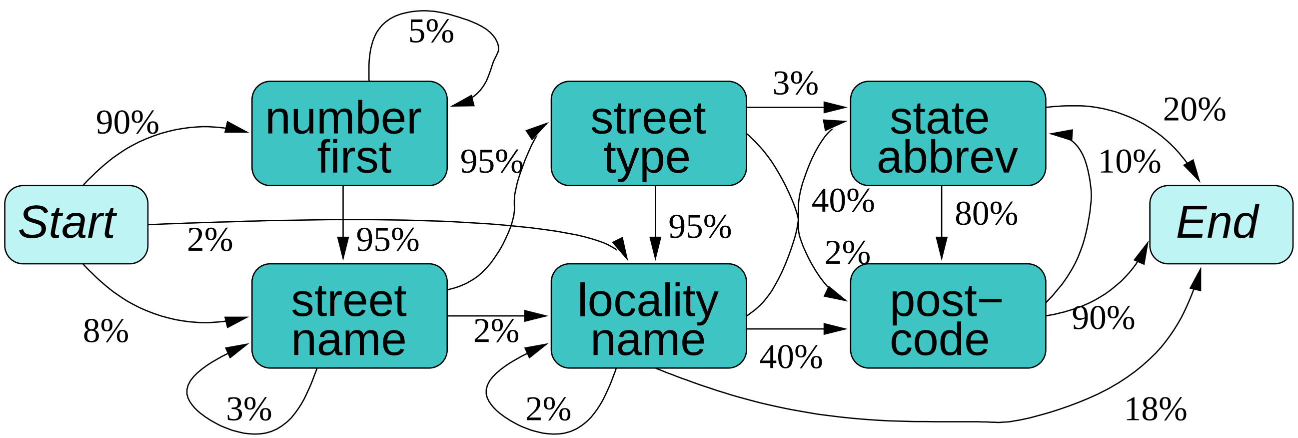

- Clean input
  - Remove unwanted characters and words
  - Expand abbreviations and correct misspellings
- Segment address into well defined *output fields*
- Verify if address (or parts of it) exists

# *Address standardisation approaches*

- Traditionally: Rules based
  - Manually developed parsing and transformation rules
  - Time consuming and complex to develop and maintain
- Recently: Probabilistic methods
  - Mainly based on *hidden Markov models* (HMMs)
  - More flexible and robust with regard to new unseen data
  - Drawback: Training data needed for most methods

*HMMs are widely used in natural language processing and speech recognition, as well as for text segmentation and information extraction.*

# *What is a Hidden Markov model?*




- A HMM is a *probabilistic* finite state machine
  - Made of a set of *states* and *transition probabilities* between these states
  - In each state an *observation symbol* is emitted with a certain probability
  - In our approach, the states correspond to *output fields*

# *Probabilistic address standardisation*

- Segmentation of Indian and US addresses (Borkar, Deshmukh, Sarawagi, 2001)
  - Hierarchical features and nested HMMs
  - Allow the integration of external hierarchical databases for improved segmentation
  - Presented results better than rules-based system *Rapier*
- Attribute recognition models (Agichtein, Ganti 2004)
  - Automatic system only using an external database
  - Based on HMMs, capture the characteristics of values in database
  - Feature hierarchies are used to learn the HMM topology and probabilities

# *Our standardisation approach*

- Based on our previous work (Churches'02)
  - Uses lexicon-based tokenisation rather than original values as HMM observation symbols
  - Manually compiled look-up tables
  - Manual preparation of training data needed
  - Better results than rule-based system *AutoStan*
- New contributions (AusDM'05)
  - Build initial HMM structure from postal guidelines
  - Automatically create HMM training data using initial HMM structure and a national address database
  - Automatically create look-up tables from address database

# *Address standardisation steps*

- Three step approach
  1. Cleaning
     – Based on look-up tables and correction lists
     – Remove unwanted characters and words
     – Correct various misspellings and abbreviations
  2. Tagging
     – Split input into a list of words, numbers and separators
     – Assign one or more tags to each element of this list (using look-up tables and/or features)
  3. Segmenting
     – Use a trained HMM to assign list elements to *output fields*

# *Tagging step*

- Tags are based on look-up tables and features
  - If found in look-up tables for street name (`SN`), street type (`ST`), locality name (`LN`), postcode (`PC`), etc.
  - Otherwise according to more general features
- Features characterise values
  - If a value contains letters (`L`), numbers (`N`), alpha-numerics (`A`), or is mixed (`M`)
  - The length of a value (`1`, `2`, ... , `6_8`, `9_11`, `12_15`, `16+`)
- Examples:

`'avenue'` will be tagged with `'ST'` and `'L6_8'`

`'2602'` will be tagged with `'PC'` and `'N4'`

`'12b'` will be tagged with `'A3'`

# *Example address standardisation*

- Raw input address: `'42 meyer Rd COOMA 2371'`
- Cleaned into: `'42 meyer road cooma 2371'`
- Tagged (both look-up tables and feature tags):

```
['42','meyer','road', 'cooma',   '2371' ]
['N2','SN/L5','ST/L4','LN/SN/L5','PC/N4']
```

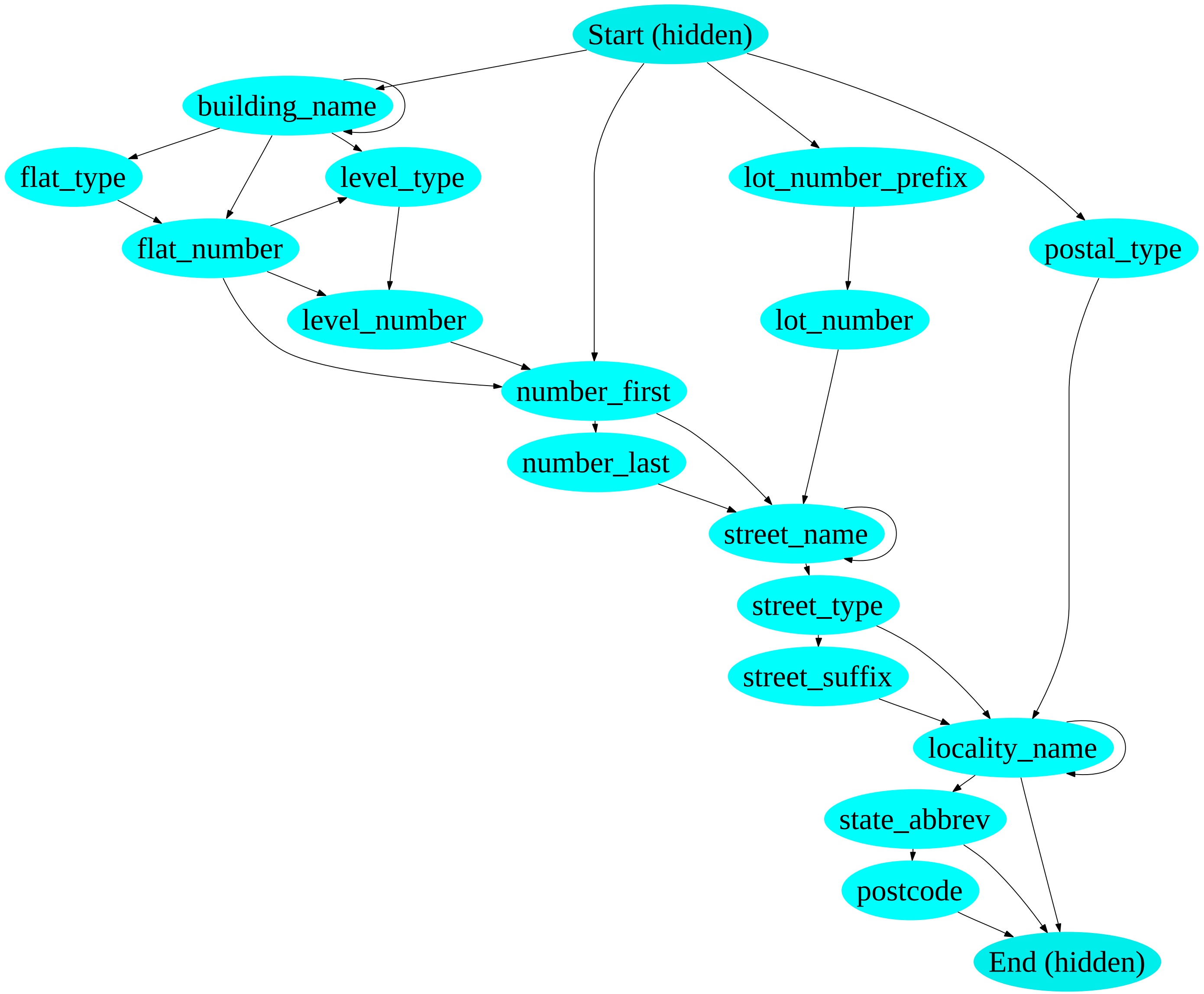
- Segmented by HMM into *output fields*:

```
 number_first :  '42'
  street_name :  'meyer'
  street_type :  'road'
locality_name :  'cooma'
     postcode :  '2371'
```

# *Preparation and training phase*

- Initial HMM structure is built using national postal guidelines (Australia Post, AS4212-1994, AS4590-1999)
  - Currently manual, in future XML scheme likely
- Records from a comprehensive address database are used as HMM training records
  - We use G-NAF (Geocoded National Address File) with around 4.5 million addresses from NSW
  - Contains clean and segmented records (26 attributes)
  - Missing are postal addresses and many postcodes, as well as characters like slash ( / ) and hyphen ( – )

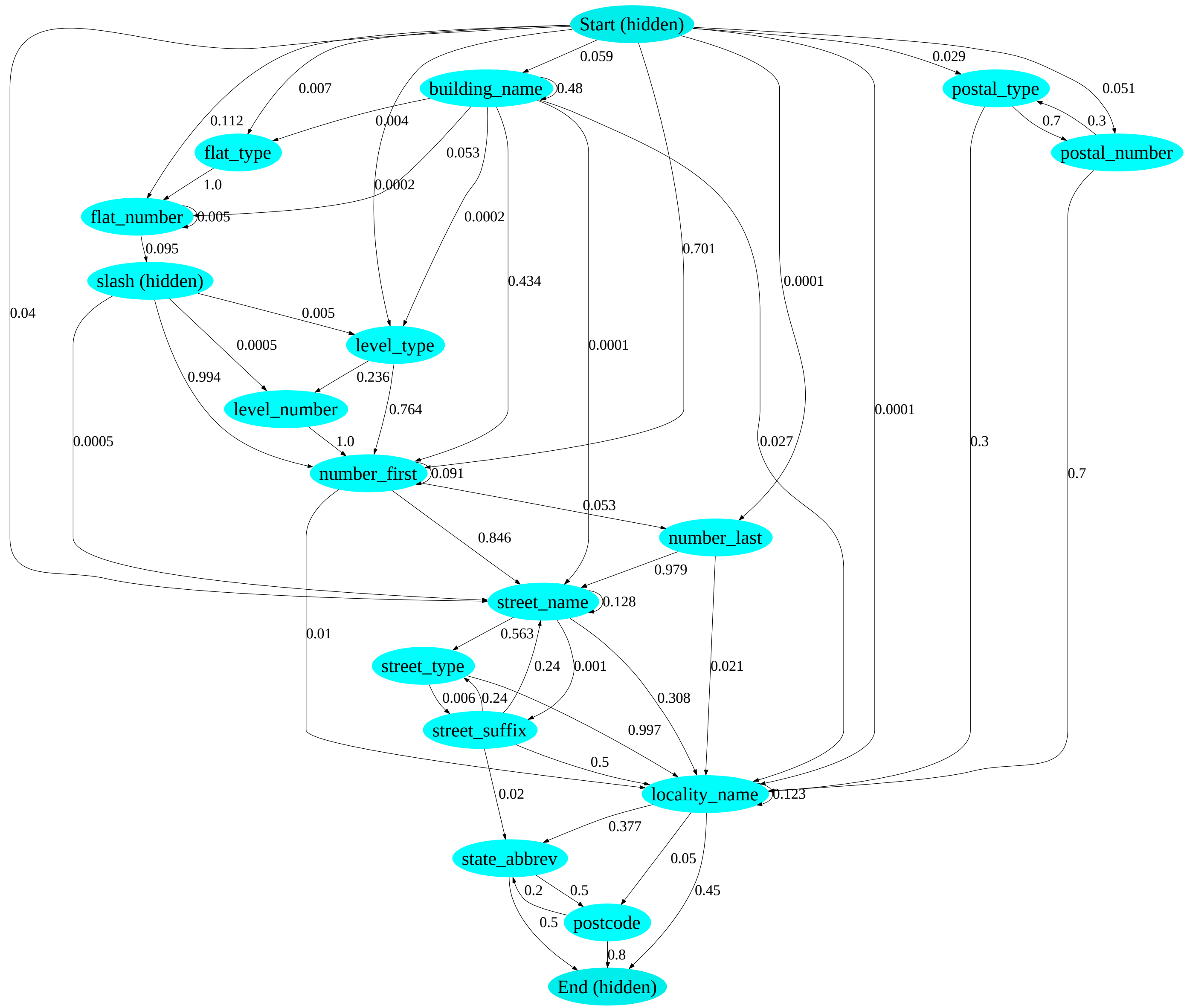
# Initial HMM structure (simplified)



# *Automated HMM training*

- Address records are re-ordered according to topologically sorted initial HMM structure
- Various tweaks need to be done
  (insert postcodes, postal addresses, slash, hyphen, etc.)
- HMM observation symbols are tags
  (either features only (**F**), look-ups only (**LT**) or both look-ups and features (**LT&F**))
- Processed records are then used for HMM training
  (smoothing is used to make HMM more robust towards unseen data in the standardisation phase)
- Look-up tables are built for name attributes
  (and merged into existing tables)

# Final HMM (simplified)






THE AUSTRALIAN NATIONAL UNIVERSITY

# *First experiments*

- Three smaller data sets
  - NSW Midwives data (500 records, randomly selected)
  - Nursing homes (600 records, randomly selected)
  - Unusual addresses (150 records, manually selected)
- HMMs generated for **F**, **LT**, and **LT&F**
- Compared to manually generated HMM using earlier *Febrl* approach (Churches'02)
- Measurements
  - Correctness: Exact and *close* standardisation accuracy
  - Number of *easy* addresses (with simple structure, like `[street num, name, type; loc, state, pc]`)

# ***Results for Midwives data collection***

| | **F** | **LT** | **LT&F** | *Febrl* |
|---|---|---|---|---|
| Easy addresses | 89.0% | 87.6% | 89.2% | 82.0% |
| Accuracy | 96.6% | 95.4% | 97.4% | 96.8% |
| *Close* accuracy | 97.0% | 97.4% | 98.0% | 97.6% |
| Time per record | 6 ms | 11 ms | 92 ms | 7 ms |

# ***Results for Nursing homes data***

| | **F** | **LT** | **LT&F** | *Febrl* |
|---|---|---|---|---|
| Easy addresses | 90.3% | 89.7% | 90.3% | 88.2% |
| Accuracy | 92.7% | 98.5% | 96.7% | 96.0% |
| *Close* accuracy | 96.5% | 98.5% | 97.8% | 98.3% |
| Time per record | 7 ms | 18 ms | 445 ms | 9 ms |

# *Results for unusual addresses*

| | **F** | **LT** | **LT&F** | *Febrl* |
|---|---|---|---|---|
| Easy addresses | 20.6% | 18.0% | 20.6% | 14.7% |
| Accuracy | 79.3% | 72.7% | 92.7% | 96.0% |
| *Close* accuracy | 80.7% | 80.0% | 94.7% | 96.0% |
| Time per record | 7 ms | 37 ms | 720 ms | 10 ms |

- 150 manually selected unusual address records (like rural addresses, corner addresses, building and institution addresses, etc.)

# *Conclusions and outlook*

- Automated address standardisation
  - Important for data mining pre-processing and linkage
  - Can be achieved using national address guidelines and a comprehensive address database
  - Accuracy comparable to hand-trained systems
- Current and future work
  - Fine-tune training data preparation
  - Add address verification (use inverted indices and hash-encodings like *MD5* or *SHA*)
  - Further testing and comparison experiments
  - Full integrate into data linkage system *Febrl*