

Reviewer's report

Title: Preparation of name and address data for record linkage using hidden Markov models

Authors:

Dr Tim Churches (tchur@doh.health.nsw.gov.au)
Peter Christen (peter.christen@anu.edu.au)
Kim Lim (klim@doh.health.nsw.gov.au)
Justin X Zhu (u3167614@student.anu.edu.au)

Version: 1 **Date:** 11 Nov 2002

Reviewer: Dr J. Michael Dean

Level of interest: A paper of considerable general medical or scientific interest

Advice on publication: Accept after discretionary revisions

This is a very interesting paper that introduces FEBRL, an open source project to develop probabilistic linkage software to support the public health community. The system is written in Python, and the authors are also pursuing efforts at parallel processing to expedite such record linkage. Currently, the public health researcher is hampered in this activity because commercial linkage software is 1) scarce; and 2) often too expensive.

The current manuscript is solely devoted to field standardization, concentrating on addresses and names. Rather than simple token matching, the authors have taken the novel approach of training hidden Markov models.

Compulsory Revisions: None

Discretionary Revisions:

1. If the reader is not already familiar with record linkage and field standardization, this will be a difficult paper. The use of hidden Markov models is novel and complex; I needed to download the FEBRL distribution, printout the documentation, and read through it before I really understood the manuscript. Even after reading this material, the current manuscript would be improved by better discussion of emission probabilities. Transition probabilities make some intuitive sense, but there really is no definition in the manuscript for emission probabilities.
2. The authors present a simplified hidden Markov model (HMM), and for teaching purposes this is terrific. However, the interesting data in this paper concerns real models, and I think they should add figures to show the HMM used for the research itself. I want to see the address model, etc.
3. Performance issues are not addressed in this paper, and are important on several levels. First, Python is interpreted and some readers might conclude it would be useless for production use. It is ideal for prototyping and method development. Second, HMM calculations potentially explode in an exponential manner, mitigated by the Viterbi algorithm and the underlying relationships in the real world,

which prevent the HMM from being ergodic (luckily!). Third, training may need to be specific for different researchers.

In each of these areas, performance data would be extremely helpful. For example, how many minutes or hours are required on commonly available hardware (single processor microcomputers) to train a fairly complex HMM (such as their production address model) as described in the paper? Next, with a trained model, how long does it take to process 10,000 records, 100,000 records, 1 million records, etc. Third, what is the impact of using C modules for the mathematical aspects in which Python is unlikely to be very efficient? If the authors have not gone this far, at least the performance numbers mentioned will allow us to know the scope of the potential problem. This would also open up areas for other open source developers to proceed as part of the community effort. If the problem is terrible, even with compiled C modules for computational bottlenecks, then this entire methodology will rely on parallel architectures. Open source or not, such architectures are not widely available.

4. What happens if a researcher does not use a bootstrap training method. Just throw 50,000 records at the trainer, equipped with reasonable look up tables. If I understand the system, if the look up tables are very comprehensive, then automatic training may be able to achieve 99.9% of the goal. This would be important for researchers who want to use the software, but would be seriously bothered by training with 100 hand coded records, followed by the bootstrap method.

It is likely that the HMM coefficients are relatively stable across numerous domains of names and addresses. What would be more likely to change are the look up tables. Or do I misunderstand?

5. Though the authors present some comparison with token based rule systems such as Autostan, performance comparisons would be important. Which method is more scaleable? One would think that Autostan performance would be roughly linear with the size of the file. While training HMM is likely to be exponential, once the model is trained with a reasonably small number of data records, and once the look up tables are comprehensive, would performance be comparable to Autostan?

In summary, this is a paper of great interest to researchers who are pursuing record linkage in the public health field. It is of even greater interest to those of us interested in open source software development in public health.

Competing interests:

None declared.