Communal Detection of Implicit Personal Identity Streams

Clifton Phua
Monash University
clifton.phua@
infotech.monash.edu.au

Ross Gayler
Baycorp Advantage
ross.gayler@
baycorpadvantage.com

Kate Smith-Miles
Deakin University
katesm@deakin.edu.au

Vincent Lee
Monash University
vincent.lee@
infotech.monash.edu.au

Abstract

The purpose of this paper is to outline some of the major developments of an identity crime/fraud stream mining system. Communal detection is about finding real communities of interest. The algorithm itself is unsupervised, single-pass, differentiates between normal and anomalous links, and mitigates the suspicion of normal links with a dynamic global whitelist. It is part of the important and novel communal detection framework introduced here for monitoring implicit personal identity streams. For each incoming identity example, it creates one of three types of single link (black, white, or anomalous) against any previous example within a set window. Subsequently, it integrates possible multiple links to produce a smoothed numeric suspicion score. In a principled stream-like fashion and using eighteen different parameter settings replicated over three large window sizes, this paper highlights and discusses significant score results from mining a few million recent credit applications.

1. Introduction

Identity crime is well-known, prevalent, and costly. It involves abusing a real person's name, address, and other identity details; usually for financial gains, without his/her explicit consent. Everyone is vulnerable to it; and the estimated cost to developed nations without nationally registered identity numbers is often in billions of dollars. Data matching against available verification databases is an existing and common practical approach. But this has data quality issues of accuracy, completeness, and timeliness in the public telephone and address directories, semi-public voters' register, and private blacklists respectively.

The task is to seek signs of identity crime within millions of streaming personal identities from credit application data *per se*. Our communal detection framework is novel as it deals specifically with the implicit and sparse nature of credit applications (the term examples is synonymous):

Implicit-ness - There are no direct interactions between identity examples. Yet, there exist underlying normal intra- and inter-personal communal relationships, as well as anomalous links which indicate illegitimate re-use of identity attribute values. Given that, the creation of implicit or pseudo-links will be a three-way trade-off among the approximate capture of normal relationships in whitelist, anomalous links outside whitelist, and the creation of redundant items in whitelist.

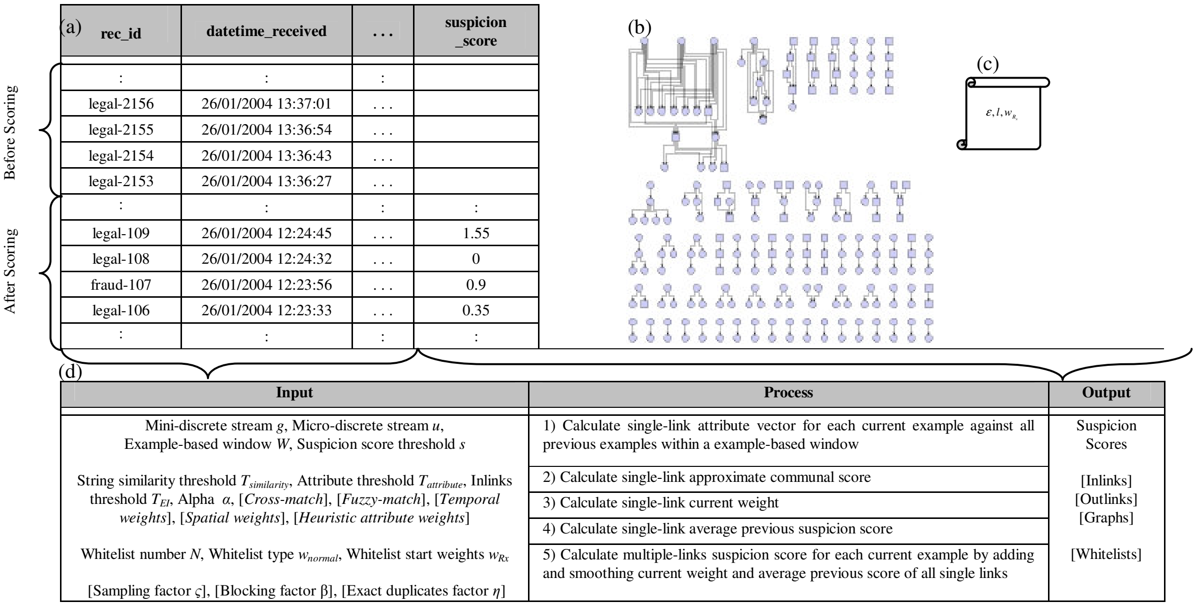
Sparsity - Each structured identity example consists of mainly text/string/identity attributes which can take on an enormous number of possible values. In fact, raw identity attribute values are so sparse/rare such that when many of them within an example become denser/frequent, the example itself becomes more interesting. Extreme sparsity is likely to cripple most clustering algorithms (too many clusters, convergence problems) and association-rule mining algorithms (too many spurious rules, low rule support).

Compared to prior work [7; 8], the current work presented here is a substantial technical improvement. It has been extended to a communal detection framework which automates the entire input (data and given parameter settings), process (the algorithm), and output (scores, graphs, and whitelists) sections. It has also dealt with the issues identified in prior work by incorporating three more parameters; performing several hundred preliminary experiments, each on a few million examples.

Section 2 is the framework overview. The complexities of streams and examples, the details of the algorithm, choice and modifications of performance metrics, are described in Sections 3, 4, 5 respectively. Section 6 presents the results and Section 7 the insights. Section 8 concludes the paper.

2. Framework overview

The communal detection framework is an original defensive fraud detection system which applies record linkage [9] techniques on data streams [1; 6]. In Figure 1(a), the input data is continuously extracted from the entire database and consists of examples sorted in descending arrival date and time order (a more recent example is placed above an older one). These extracted examples are either previously scored (previous stream which exists within the window), or the ones to be scored in arrival order (the current stream). The last column in Figure 1(a) contains the suspicion score output. The examples and links in Figure 1(b) and the whitelist in Figure 1(c) are the visual outputs for a current stream.



	rec_id	datetime_received	. . .	suspicion _score
Before Scoring	:	:	:	
	legal-2156	26/01/2004 13:37:01	. . .	
	legal-2155	26/01/2004 13:36:54	. . .	
	legal-2154	26/01/2004 13:36:43	. . .	
	legal-2153	26/01/2004 13:36:27	. . .	
After Scoring	:	:	:	:
	legal-109	26/01/2004 12:24:45	. . .	1.55
	legal-108	26/01/2004 12:24:32	. . .	0
	fraud-107	26/01/2004 12:23:56	. . .	0.9
	legal-106	26/01/2004 12:23:33	. . .	0.35
	:	:	:	:

Input	Process	Output
Mini-discrete stream *g*, Micro-discrete stream *u*, Example-based window *W*, Suspicion score threshold *s* String similarity threshold $T_{similarity}$, Attribute threshold $T_{attribute}$, Inlinks threshold T_{EI}, Alpha α, [*Cross-match*], [*Fuzzy-match*], [*Temporal weights*], [*Spatial weights*], [*Heuristic attribute weights*] Whitelist number *N*, Whitelist type w_{normal}, Whitelist start weights w_{Rx} [Sampling factor ς], [Blocking factor β], [Exact duplicates factor *η*]	1) Calculate single-link attribute vector for each current example against all previous examples within a example-based window 2) Calculate single-link approximate communal score 3) Calculate single-link current weight 4) Calculate single-link average previous suspicion score 5) Calculate multiple-links suspicion score for each current example by adding and smoothing current weight and average previous score of all single links	Suspicion Scores [Inlinks] [Outlinks] [Graphs] [Whitelists]

Figure 1: Communal detection framework overview of input, process, and output sections
* square brackets indicate that enclosed parameter is optional

In Figure 1(d)'s "Input" column, the first four parameters determine the amount of input data to continuously extract with Structured Query Language (SQL) queries from the database to process an incoming stream. The next nine parameters are used to fine-tune the processing capabilities of the algorithm. The following three parameters determine the next whitelist's contents. The last three parameters are meant to speed up the experiments to get estimated results. The "Process" column has five main calculation/algorithmic steps to generate a suspicion score for each example. Simply put, the algorithm is an unsupervised, single-pass approach which requires raw and unlabelled data; and operates on a three-list principle: blacklist, whitelist and greylist to score a current example for its relative density. For the "Output" column, this paper evaluates the suspicion scores (the higher the score for each credit application, the higher the suspicion/risk) and visualisations.

3. Data stream layers

In typical graph theory notation, the simple directed graph can be described as having two sets as $G = \langle V, E \rangle$. V represents a set of vertices (examples) and E represents a set of directed edges (links). For processing of implicit personal identity streams in the real world, let G represent a *rapid* and *continuous* overall stream with an ordered set of $\{..., g_{x-2}, g_{x-1}, g_x, g_{x+1}, g_{x+2}, ...\}$ *mini-discrete* streams. Let g_x represent the current mini-discrete stream with an ordered set of $\{u_{x,1}, u_{x,2}, ..., u_{x,p}\}$ *micro-discrete* streams, where p is the variable total number of micro-discrete streams. Each g_x is a container of multiple micro-discrete streams in order to hold a certain volume of historical data, and to create a new whitelist for g_{x+1}.

Let $u_{x,y}$ represent the current micro-discrete stream within the current mini-discrete stream with an ordered set of $\{v_{x,y,1}, v_{x,y,2}, ..., v_{x,y,q}\}$ examples to be processed online, where q is the variable total number of examples. Each $u_{x,y}$ is basically a snapshot of recent examples. For simplicity, the subscripts x and y will be omitted as processing is always carried out on q examples and from the viewpoint of the most recent micro-discrete stream $u_{x,y}$'s oldest unscored example $v_{x,y,1}$ to the newest unscored example $v_{x,y,q}$.

In our problem, the overall simulated stream from a credit bureau-contributed dataset is more than one year's worth of real, labelled (as "fraud" or "legal"), recent, consecutive, and time-stamped credit applications, numbering more than a few million (the experiments in this paper use slightly more than half the data). On average, for each month and each day, a few hundred thousand applications and more than ten thousand applications respectively stream into the central database. Only less than one percent of these are known to be fraudulent. For experimental purposes here, each g and each u represents each month's applications and at least one day's worth of applications respectively.

Every original example contains sparse identity attributes such as personal names, addresses, telephone numbers, driver licence numbers (or social security numbers), date-of-births, and other personal identifiers. Privacy and confidentiality are enforced by hashing a few key personal identifiers to an irreversible ten-digit number, although it means that fuzzy matching cannot be performed on the hashed attributes. With this ethical and legal safeguard, the actual identity in each example cannot be ascertained from the dataset-on-hand. In total, each example consists of about thirty raw attributes and with most of them required by the algorithm to produce a numeric suspicion score.

4. Algorithm design

Let v_i each represent a current unscored example (credit application) and labelled with a set of N attributes respectively, where $v_i = \{a_{i,1}, a_{i,2}, ..., a_{i,N}\}$.

Each v_i is compared against an example-based window W of previously scored examples $\{v_{i-1}, v_{i-2}, ..., v_{i-W}\}$. Depending on order of v_i, these are consecutively scored examples within u, and/or a different set of historically scored examples u_s sorted in descending order of suspicion score $S(v_i)$. The suspicion score threshold s can be used to filter the least suspicious examples within u_s. Here, the results of two SQL queries are merged into a single table in memory.

For simplicity, v_j is used to denote a previously scored example. Let $e_{i,j}$ represent the directional link (relationships between examples).

4.1 Attribute vector

$$\varepsilon[v_i, v_j] \\ = \{\varepsilon_1[a_{i,1}, a_{j1}], \varepsilon_2[a_{i,2}, a_{j,2}], ..., \varepsilon_N[a_{i,N}, a_{j,N}]\}, \forall i, j$$

where ε is the attribute vector, between v_i and v_j, which represents the relationship for an example-pair. ε_k is case sensitive and can also be optionally determined by $\varepsilon_k[a_{i,k}, a_{j,l}]$, where k is cross-matched with l (another similar attribute).

$$S(\varepsilon_k) = \begin{cases} 1 & \text{if Simmetrics}(a_{i,k}, a_{j,k}) \geq T_{similarity} \\ 0 & \text{otherwise} \end{cases}$$

where $S(\varepsilon_k)$ is the Boolean suspicion score of each pair-wise attribute ε_k. Simmetrics(.) is a library of string similarity metrics [4]. For all experiments here, a fast metric *Jaro-Winkler* is used. $T_{similarity}$ is the threshold for fuzzy matching and $0 \leq T_{similarity} \leq 1$ where 0 denotes totally different and 1 as an exact match.

$$S(\varepsilon) = \begin{cases} \sum_{k=1}^{N} [S(\varepsilon_k) * w_{attribute_k}] & \text{if } \sum_{k=1}^{N} S(\varepsilon_k) \geq T_{attribute} \\ 0 & \text{otherwise} \end{cases}.$$

where $S(\varepsilon)$ is the attribute vector score and $0 \leq S(\varepsilon) \leq 1$. Each pair-wise attribute weight $w_{attribute_k}$ can be equal by default or can be optionally assigned such that $\sum_{k=1}^{N} w_{attribute_k} = 1$. In order to link any example-pair, $T_{attribute}$ is the threshold for the minimum number of matched attributes required to link v_i to v_j for a score and $N \geq T_{attribute}$.

4.2 Approximate communal score

$$w_{communal_{i,j}} \\ = \begin{cases} 1 \,\&\, v_i \xrightarrow{fraud} v_j & \text{if } v_j = \text{fraud} \,\&\, S(\varepsilon) > 0 \\ [w_{R_x} * S(\varepsilon)] \,\&\, v_i \xrightarrow{normal} v_j & \text{if } \varepsilon \in \Re \,\&\, S(\varepsilon) > 0 \\ S(\varepsilon) \,\&\, v_i \xrightarrow{anomalous} v_j & \text{if } \varepsilon \notin \Re \,\&\, S(\varepsilon) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where each of the first three conditions constructs a blacklist, whitelist, and greylist score and link respectively. $w_{communal_{i,j}}$ is the communal link weight derived from pair-wise matching of identifier/string attributes (numerical attributes are treated as identifier/string attributes) and $0 \leq w_{communal_{i,j}} \leq 1$. The origins, purpose, description, and construction of a whitelist $\Re$ are discussed further in sub-section 5.6.

4.3 Current weight

$$w_{i,j} \\ = \begin{cases} w_{communal_{i,j}} * w_{temporal_{i,j}} * w_{spatial_{i,j}} & \text{if } w_{communal_{i,j}} > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $w_{i,j}$ is the overall current link weight between v_i and v_j and roughly captures the strength of the communal links over time and space and $0 \leq w_{i,j} \leq 1$. $w_{temporal_{i,j}}, \forall i, j$ is the optional link weight derived from pair-wise time difference (in days) of *datetime_received* and $0.5 \leq w_{temporal_{i,j}} \leq 1$; $w_{spatial_{i,j}}, \forall i, j$ is the optional link weight derived from pair-wise geographical distance (in kilometres) of *postcode* and $0.5 \leq w_{spatial_{i,j}} \leq 1$.

4.4 Average previous score

$$S(v_j)/E_O(v_j) \\ = \begin{cases} 1 & \text{if } w_{i,j} > 0 \,\&\, \{[v_i \xrightarrow{fraud} v_j] \\ & \text{or } [(v_i \xrightarrow{normal} v_j \text{ or } v_i \xrightarrow{anomalous} v_j) \\ & \&\, E_I(v_j) \geq T_{E_I}]\} \\ S(v_j)/E_O(v_j) & \text{if } w_{i,j} > 0 \,\&\, S(v_j) > 0 \,\&\, E_O(v_j) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$S(v_j)$ is the total suspicion score of a previous example which v_i links to. $E_O(v_j)$ is the number of outgoing links from v_j. Therefore, $S(v_j)/E_O(v_j)$ is the average suspicion score of each previous example and $0 \leq S(v_j)/E_O(v_j) \leq 1$. $E_I(v_j)$ is the number of incoming links into v_j. T_{E_I} is the threshold for the acceptable number of incoming links and $0 \leq T_{E_I} < W$.

4.5 Suspicion score

$$S(v_i) = \sum_{v_j \in M(v_i)} \left[w_{i,j} + S(v_j) / E_O(v_j) \right]$$

where $S(v_i)$ is the total suspicion score of the current credit application v_i with k -pairs (multiple-pairs) of linked v_j and $S(v_i) \geq 0$. $M(v_i)$ is the set of k examples which v_i links to. Therefore, a high suspicion score $S(v_i)$ indicates three possibilities: strong connection between v_i and v_j (high link weight - $w_{i,j}$) which is dependent on number of non-missing attribute values; and/or the high quality of v_j (high average suspicion scores - $S(v_j)/E_O(v_j)$); and/or the large quantity of v_j (many connected examples - k).

$$S(v_i) = \sum_{v_j \in M(v_i)} \left[(1-\alpha) * w_{i,j} + \alpha * S(v_j) / E_O(v_j) \right]$$

where α is exponential smoothing factor needed to gradually discount the effects of previous suspicion scores and $0 \leq \alpha \leq 1$.

4.6 Whitelist construction

Good quality whitelists are critical to the entire communal detection framework. They are the novel amalgamation of ideas from whitelist-based spam filtering, social network analysis [5; 10], and outlier detection. According to their definitions, a whitelist is a global list of accepted items within a set (opposite of a blacklist) which allows the limited explanation and measurement of human relationships. Also, a non-whitelist item can be regarded as one which is so different from the whitelist items that it was produced artificially.

Our link whitelist consists of a finite N number of ranked relationship-types between examples, instead of an absolute whitelist which can be made up of the actual examples. Its main purpose is to reduce the suspicion scores and incidence of false positives, while maintaining true positives. Right after the end of g_x , a completely new whitelist is constructed using $\varepsilon \in g_x$ for g_{x+1} . This is based on the assumption that the legitimate items make up a significant proportion of the total number of items and these legitimate items change over time. Therefore, the following are three important and related questions to produce a good quality whitelist: What should each item consist of? What is the ideal number of whitelist items? What are the changes between successive whitelists?

Each item is a unique relationship-type and must at least be made up of pair-wise identifier/string attributes ε , item volume l , and item weight w_{R_x} . Some items which reflect explicit anomalous relationships are automatically excluded during the whitelist construction.

There is no magic number but there are three justifications to rank and keep only the items with largest volume (usually the larger the volume, the smaller the weight) in the whitelist. First, it captures a wide range of common social known or unknown relationships - from casual acquaintances to tight familial bonds; and also spurious links which can increase significantly under certain parameter settings. Second, non-whitelist items are actually rarer links which have been excluded from the whitelist and penalised with higher scores. Third, with a shortened whitelist, the number of comparisons of each incoming link to items is reduced.

The whitelist changes artificially with different attribute- and whitelist-related parameters, and legal behaviour evolves naturally over time. Whitelist measurements of parameter changes and legal behaviour evolution involves tracking changes in items, item volume, existing item rank, and existing item volume between consecutive whitelists, relative to the number of whitelist items N .

For concreteness, let $\Re = [R_1, R_2, ..., R_N]$ be a set of one or more selected relationship-types defined as normal where each normal relationship R_x is ranked in descending order by (or filtered according to) item volume $l \in L$. Let $w_{normal} \equiv [w_{R_1}, w_{R_2}, ..., w_{R_N}]$ where w_{R_x} is the corresponding item weight of R_x and $0 \leq w_{R_x} \leq 1$. w_{normal} is sorted in ascending order, and there are three main types of weights “a”, “b”, and “c”: constant (all weights are w_{R_1}), rank-based and frequency-based where each subsequent weight is incrementally increased by $[(1 - w_{R_1})/N]$ and $[(1 - w_{R_1}) * (l/L)]$ respectively.

Each attribute vector comprises of $\varepsilon = \varepsilon_1 \,\&\, \varepsilon_2 \,\&\, ... \,\&\, \varepsilon_N$, and each pair-wise identifier/string attribute can have one of four types of string values $\varepsilon_k = ["O","Y","N","?"]$. “O” means attribute is deliberately omitted, “Y” represents a pair-wise attribute match, “N” a non-match, and “?” indicates one/both missing attribute value(s).

In summary, the whitelist captures pairs or groups of real people who share common personal identifiers to reduce $S(v_j)$. However, its usefulness depends on the empirical choice and optimum combination of the attribute- and whitelist-related parameters.

4.7 Data quality improvement

The data quality problems are found at the v_i , v_j , and ε levels; and predominantly are by-products of hashing, testing, and submission errors. To understand the experimental results in a practical setting, our approach is to filter them as rapidly as possible during processing. To speed up experiments, there is one heuristic technique for each of the three levels.

At the v_i level, a few actual sub-streams, comprising slightly more than 10% of all examples, are filtered because some of their unstructured attributes were hashed into just one value. Several dummy sub-streams, comprising less than 1.5% of all examples, are particularly common in some months and must be filtered. For increased speed, a sampling factor ς is for keeping all minority class; randomly under-sampling the dominant class (legal examples) and $0 \leq \varsigma \leq 1$ where 0 denotes no filtering and 1 as filtering all legal examples. For this research, ς is to approximately ascertain robust global properties of the data; empirically, in model-building, ς can be used as load shedding during bursty input streams.

At the v_j level, in addition to filtering the few actual and all the dummy sub-streams, if either pair-wise attribute values $a_{i,k}$ or $a_{j,k}$ consist of ten zeros (null attribute value which has been hashed) or is missing, then it is reflected as $\varepsilon_k = "?"$. For increased speed, a Boolean blocking factor β [2] is for filtering v_j immediately when the blocking key(s) is not the same as v_i. If the key(s) is the same, the rest of the attributes will be compared. In the algorithm, blocking is performed on either of two attributes and these are chosen because they are least likely to be manipulated, have the least missing values, spelling, and transcription errors compared to the rest.

At the ε level, to compute $w_{communal_{i,j}}$ and $S(v_j)/E_O(v_j)$ practically, v_j is only considered a known fraud if its fraud-status entry date and time is earlier than v_i. For increased speed, an exact duplicate factor η is for removing links of exact duplicates from the same sub-stream within a pair-wise time difference (in minutes) and $0 \leq \eta \leq \infty$ where 0 filters nothing and a large number filters most, if not all, exact duplicate links. Exact duplicates from same sub-stream within a short time frame are likely to be the result of errors and these can contaminate the whitelist.

5. Performance measures

The score distribution is heavily skewed to the left and each score can exceed 1. Since zero scores are not going to be investigated, they are not relevant to the decision-making process. Excluding them will result in more realistic curves, although the number of examples in each curve will most likely be different.

Without actual cost figures, FME' curves, NTOP-*k* [3] and AUC' measures are the most appropriate for the scores. These four score evaluation metrics are described, and their strengths and weaknesses are listed in Table 1.

Table 1: Evaluation criteria for scores

Metric	Description	Strengths	Weaknesses
FME' curve	FME' is plotted against eleven threshold values.	Multiple-values, performance under different thresholds, zeros are excluded.	-
NTOP-*k*	All scores are ranked in descending order; PPV in top *k* percentage of scores.	Single-value, easiest to interpret, concerned only with most suspicious, zeros are naturally	A form of threshold-setting.
AUC'	Area under the ROC curve.	Single-value, easy to interpret, zeros are excluded.	Designed for only standard classification algorithms

6. Experiments

The next two sub-sections outline the experimental design and score results.

6.1 Design

Based on previous experiments, the first strategy here is to have three sets of experiments. Each progressive set utilises a larger W which comprises tens of thousands of examples: set a is half the W of set b and a quarter of set c. Intuitively, a larger W improves the results, but at the same time there is a performance bottleneck due to the comparison of attributes between more pairs of examples [9].

The second strategy is to start with the best known parameter setting (baseline experiment), observe if there is any significant improvement by holding all parameter values constant and tuning to one promising parameter value. The framework extracted streaming examples using $g =$ a few hundred thousand examples, $u = W, s >= 0.001$. The algorithm processed with $T_{similarity} = 0.8, T_{\text{attribute}} = 3, T_{E_I} = 10, \alpha = 0.8$ and with *Cross-match* = *TRUE*, *Fuzzy-match* = *TRUE*. The whitelist is configured at $N = 100, w_{normal} = "b", w_{\mathrm{R}_1} = 0.01$, and experimental speed-up is achieved with $\varsigma = 0.9, \beta = TRUE, \eta = 120$. For the fifty-four experiments, $T_{similarity}, \varsigma, \beta$ are fixed throughout. Table 2 lists three sets $\{a, b, c\}$ of attribute-related experiments 1 to 8 and 14 to 18; and whitelist-related experiments 9 to 13.

Table 2: Experiments/parameter settings

Set	Experiment	Parameter Change(s)
	1	-
a	**2**	*Cross-match = FALSE, Fuzzy-match = FALSE*
	3	*Temporal weights = TRUE, Spatial weights = TRUE*
	4	$\alpha = 0.5$
b	5	$T_{E_I} = 4$
	6, **7**, 8	$T_{attribute} = 2, T_{attribute} = 4, T_{attribute} = 7$
	9, 10	$N = 0, N = 200$
c	11, 12	$w_{normal} = "a", w_{normal} = "c"$
	13	$w_{\mathrm{R}_1} = 0.5$
	14, 15	$\eta = 0, \eta = 2880$
	16, 17	$s >= 0, s >= 0.1$
	18	*Heuristic attribute weights = TRUE*

6.2 Score results

In figure 2, the best FME' curves are experiments 7, 2, 8 and 18. In figure 2(a), a8 is slightly better than a7 by 0.004, significantly better than a2 by 0.02 at threshold 0.2. But in figure 2(b), b7 rose by 0.022, b2 by 0.001, and b8 fell by more than 0.04. In figure 2(c), c7 and c2 are significantly higher than the next nearest curve by at least 0.04. The poorest FME' curves are also the computationally most expensive experiments 6, 16, and 3, with none of these peaking above 0.09. Figure 2 demonstrates two points. First, with the right threshold and a larger W, two separate parameter changes can exceed the baseline by a significant FME'. Second, computationally cheaper parameter settings can give much better results.

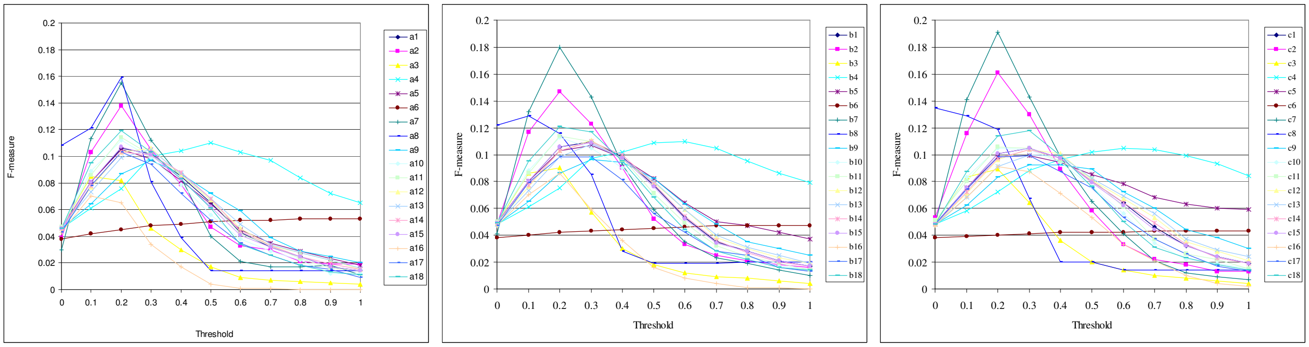


Figure 2: FME' curves of **(a):** set a, **(b):** set b, **(c):** set c, of all experiments (curves)

Interestingly, figure 3(a) shows that with NTOP-1 and across three W sizes, the best experiment is 18, followed by 2 and 7 (c7 has the best NTOP-1 value). Figure 3(b) confirms that with AUC', the best experiments are 7, 2, 11, and 18. From figures 3 and 4, experiment 1 highlights the importance of communal detection with the global whitelist by consistently outperforming 9 (no whitelist); losing out to 10 (2 * N whitelist items) and 11 ($w_{R_x} = w_{R_1}$).

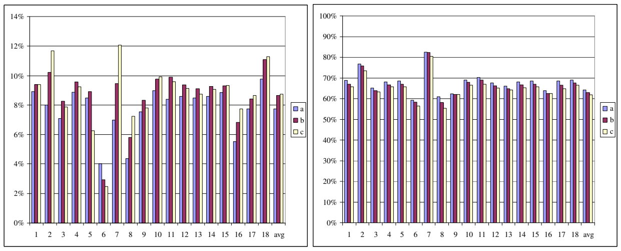


Figure 3(a): NTOP-1 results, **Figure 3(b):** AUC' results, of three sets with averages (coloured bars), all experiments (*x*-axes), of all sub-streams

7. Discussion and limitations

Through these current sets of experiments, the better parameter settings do confirm that quality whitelists are essential, certain parameters ($T_{attribute}$, *Cross-match*, *Fuzzy-match*) and attributes (*Heuristic attribute weights*) are more important than others, and the need to improve data quality (ς , β , and η). On the other hand, it is recognised that *temporal/spatial weights* are not useful.

8. Conclusion

With no explicit link information, our approach is to find streaming approximate duplicates and represent them with implicit links. The whitelist is constructed to hold implicit link-types which reflect genuine communal relationships and reduce their suspicion score. This way, false alarms are kept to a minimum. Current results are encouraging with further details in the form of graph, whitelist, monthly and individual subscriber results to be presented.

Acknowledgements

The authors are grateful to the intellectual discussions provided by the tutorial speakers at the PAKDD06 and EII05 Schools, technical resources made available by Paul van Haaster, and financial support by the ARC under Linkage Grant Number LP0454077.

REFERENCES

[1] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom, "STREAM: The Stanford Stream Data Manager Demonstration description", *Proc. of SIGMOD03*, 2003.

[2] R. Baxter, P. Christen, and T. Churches, "A Comparison of Fast Blocking Methods for Record Linkage", *Proc. of SIGKDD03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003, pp25-27.

[3] R. Caruana, and A. Niculescu-Mizil, "Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria", *Proc. of SIGKDD04*, 2004, pp69-78.

[4] S. Chapman, Simmetrics – Open Source Similarity Measure Library. Accessed from: http://sourceforge.net/projects/ simmetrics/. Accessed in April 2005.

[5] C. Cortes, D. Pregibon, and C. Volinsky, "Communities of Interest", *Proc. of IDA2001*, 2001, pp105-114.

[6] J. Kleinberg, "Temporal Dynamics of On-Line Information Streams", In M. Garofalakis, J. Gehrke, and R. Rastogi (eds.), *Data Stream Management: Processing High-Speed Data Streams*, Springer, 2005.

[7] C. Phua, R. Gayler, V. Lee, and K. Smith, "On the Communal Analysis Suspicion Scoring for Identity Crime in Streaming Credit Applications", *European Journal of Operational Research*, 2006, submitted.

[8] C. Phua, R. Gayler, V. Lee, and K. Smith, "On the Approximate Communal Fraud Scoring of Credit Applications", *Proc. of CSCCIX*, 2005.

[9] P. Christen, T. Churches, and M. Hegland, "A Parallel Open Source Data Linkage System", *Proc. of PAKDD04*, 2004.

[10] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. Cambridge University Press, New York, 1994.