**Resilient Identity Crime Detection**

**Abstract**

Identity crime is well known, prevalent, and costly; and credit application fraud is a specific case of identity crime. The existing non data mining detection system of business rules and scorecards, and known fraud matching have limitations. To address these limitations and combat identity crime in real time, this paper proposes a new multilayered detection system complemented with two additional layers: communal detection (CD) and spike detection (SD). CD finds real social relationships to reduce the suspicion score, and is tamper resistant to synthetic social relationships. It is the whitelist-oriented approach on a fixed set of attributes. SD finds spikes in duplicates to increase the suspicion score, and is probe-resistant for attributes. It is the attribute-oriented approach on a variable-size set of attributes. Together, CD and SD can detect more types of attacks, better account for changing legal behavior, and remove the redundant attributes. Experiments were carried out on CD and SD with several million real credit applications. Results on the data support the hypothesis that successful credit application fraud patterns are sudden and exhibit sharp spikes in duplicates. Although this research is specific to credit application fraud detection, the concept of resilience, together with adaptivity and quality data discussed in the paper, are general to the design, implementation, and evaluation of all detection systems.

**Literature Review**

**R. Bolton and D. Hand, “Unsupervised Profiling Methods for Fraud Detection,” Statistical Science, vol. 17, no. 3, pp. 235-255, 2001**

Credit card fraud falls broadly into two categories: behavioural fraud and application fraud. Application fraud occurs when individuals obtain new credit cards from issuing companies using false personal information and then spend as much as possible in a short space of time. However, most credit card fraud is behavioural and occurs when details of legitimate cards have been obtained fraudulently and sales are made on a 'Cardholder Not Present' basis. These sales include telephone sales and e-commerce transactions where only the card details are required.

This paper concerned with detecting behavioural fraud through the analysis of longitudinal data. These data usually consist of credit card transactions over time, but can include other variables, both static and longitudinal. Statistical methods for fraud detection are often classification (supervised) methods that discriminate between known fraudulent and non-fraudulent transactions; however, these methods rely on accurate identification of fraudulent transactions in historical databases – information that is often in short supply or non-existent.

Unsupervised fraud detection methods *have* been researched in the detection of computer intrusion (hacking). Here profiles are trained on the combinations of commands that a user uses most frequently in their account. If a hacker gains illegal access to the account then their intrusion is detected by the presence of sequences of commands that are not in the profile of commands typed by the legitimate user.

**P. Christen and K. Goiser, “Quality and Complexity Measures for Data Linkage and Deduplication,” Quality Measures in Data Mining, F. Guillet and H. Hamilton, eds., vol. 43, Springer, 2007, doi: 10.1007/978-3-540-44918-8**

Deduplicating one data set or linking several data sets is increasingly important tasks in the data preparation steps of many data mining projects. The aim of such linkages is to match all records relating to the same entity. Different measures have been used to characterize the quality and complexity of data linkage algorithms, and several new metrics have been proposed. Data linkage and deduplication techniques have traditionally been used in the health sector for cleaning and compiling data sets for longitudinal or other epidemiological studies and in statistics for linking census and related data. As most real-world data collections contain noisy, incomplete and incorrectly formatted information, data cleaning and standardisation are important preprocessing steps for successful data linkage, or before data can be loaded into data warehouses or used for further analysis. Data may be recorded or captured in various, possibly obsolete formats and data items may be missing, out of date, or contain errors. The cleaning and standardization of names and addresses is especially important to make sure that no misleading or redundant information is introduced (e.g. duplicate records). Data linkage and deduplication are important steps in the pre-processing phase of many data mining projects, and also important for improving data quality before data is loaded into data warehouses. An overview of data linkage techniques has been presented in this chapter, and the issues involved in measuring both the quality and complexity of linkage algorithms have been discussed. It is recommended that the quality be measured using the precision-recall or F-measure graphs (over a varying threshold) rather than single numerical values, and that quality measures that include the number of true negative matches should not be used due to their large number in the space of record pair comparisons. When publishing empirical studies researchers should aim to use non-blocked data sets if possible, or otherwise at least report measures that quantify the effects of the blocking process.

**O. Kursun, A. Koufakou, B. Chen, M. Georgiopoulos, K. Reynolds, and R. Eaglin, “A Dictionary-Based Approach to Fast and Accurate Name Matching in Large Law Enforcement Databases,” Proc. IEEE Int’l Conf. Intelligence and Security Informatics (ISI ’06), pp. 72-82, 2006, doi: 10.1007/11760146**

Dirty data is a necessary evil in large databases. Large databases are prevalent in a variety of application fields such as homeland security, medical, among others. In that case, a search for specific information by a standard query fails to return all the relevant records. The existing methods for fuzzy name matching attain variable levels of success related to performance measures, such as speed, accuracy, consistency of query return times (robustness), scalability, storage, and even ease of implementation.

Name searching methods using name-by-name comparisons by edit distance (i.e., the minimum number of single characters that need to be inserted into, deleted from, and/or substituted in one string to get another) throughout the entire database render the desired accuracy, but they exhibit high complexity of run time and thus are non-scalable to large databases. In this paper, we have introduced a method (PREFIX) that is capable of an exhaustive edit-distance search at high speed, at the expense of some additional storage for a prefix-dictionary tree constructed. We have also introduced a simple extension to it, called ANSWER that has run-time complexity comparable to soundex methods, and it maintains robustness and scalability, as well as a comparable level of accuracy compared to an exhaustive edit distance search. ANSWER has been tested, and its advantages have been verified, on real data from a law-enforcement database (FINDER).

**J. Neville, O. Simsek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg, “Using Relational Knowledge Discovery to Prevent Securities Fraud,” Proc. 11th ACM SIGKDD Int’l Conf. Knowledge Discovery in Data Mining (KDD ’05), 2005**

This paper describes an application of relational knowledge discovery to a key regulatory mission of the National Association of Securities Dealers (NASD). NASD is the world’s largest private-sector securities regulator, with responsibility for preventing and discovering misconduct among securities brokers. Our goal was to help focus NASD’s limited regulatory resources on the brokers who are most likely to engage in securities violations. Using statistical relational learning algorithms, we developed models that rank brokers with respect to the probability that they would commit a serious violation of securities regulations in the near future.

NASD generates a list of higher-risk brokers (HRB) using a set of handcrafted rules they have formed using their domain knowledge and experience. This approach has two weaknesses we aim to address. First, the handcrafted rules simply categorize the brokers as “higherrisk” and “lower-risk” rather than providing a risk-ordered ranking. A ranking would be more useful to examiners as it would allow them to focus their attention on brokers considered to have the highest risk. Second, NASD’s handcrafted rules use only information intrinsic to the brokers. In other words, they do not utilize relational context information such as the conduct of past and current coworkers. NASD experts believe that organizational relationships can play an important role in predicting serious violations. For example, brokers that have had serious violations in the

past may influence their coworkers to participate in future schemes. Furthermore, some firms tend to be associated with continuous misconduct (i.e., they do not regulate their own employees and may even encourage violations). Lastly, higher-risk brokers sometimes move from one firm to another collectively, operating in clusters, which heightens the chance of regulatory problems. A model that is able to use relational context information has the potential to capture these types of behavior and provide more accurate predictions.

**R. Wheeler and S. Aitken, “Multiple Algorithms for Fraud Detection,” Knowledge-Based Systems, vol. 13, no. 3, pp. 93-99, 2000**

This paper describes an application of Case-Based Reasoning to the problem of reducing the number of final-line fraud investigations in the credit approval process. The performance of a suite of algorithms which are applied in combination to determine a diagnosis from a set of retrieved cases is reported. An adaptive diagnosis algorithm combining several neighborhood based and probabilistic algorithms was found to have the best performance, and these results indicate that an adaptive solution can provide fraud filtering and case ordering functions for reducing the number of final-line fraud investigations necessary.

Statistical investigations of the test sets suggested that the nature of the problem is inherently non-linear, noisy, contradictory, and not addressable using a simple similarity matrix and CBR decision system. This is unsurprising as the test sets were composed of the most difficult and intractable sub-set of the credit approval data, and as such did not cluster into identifiable fraud/non-fraud regions. However, highly localised phenomena and patterns appeared fairly common, suggesting that a hybrid or adaptive system within a CBR methodological structure might be able to focus upon and effectively exploit these characteristics. The proof-of-concept system design has two essential decision-making components familiar to all CBR frameworks: retrieval and diagnosis. Retrieval utilizes a weight matrix and nearest neighbour algorithm, while diagnosis utilises a suite of algorithms which analyse the data recalled by the retrieval mechanism as being significant.

**Existing System**

Existing defense is made up of business rules and scorecards. Scorecards for credit scoring can catch a small percentage of fraud which does not look creditworthy; but it also removes outlier applications which have a higher probability of being fraudulent.

Existing defense is known fraud matching. Known frauds are complete applications which were confirmed to have the intent to defraud and usually periodically recorded into a blacklist. Subsequently, the current applications are matched against the blacklist. This has the benefit and clarity of hindsight because patterns often repeat themselves.

Case-based reasoning (CBR) is the only known prior publication in the screening of credit applications. CBR analyzes the hardest cases which have been misclassified by existing methods and techniques. Retrieval uses thresholded nearest neighbor matching. Diagnosis utilizes multiple selection criteria (probabilistic curve, best match, negative selection, density selection, and default) and resolution strategies (sequential resolution-default, best guess, and combined confidence) to analyze the retrieved cases.

**Draw back**

They are untimely due to long time delays, in days or months, for fraud to reveal it, and be reported and recorded. This provides a window of opportunity for fraudsters.

Recording of frauds is highly manual. This means known frauds can be incorrect, expensive, and difficult to obtain and have the potential of breaching privacy.

**Proposed System**

This paper proposes crime detection in credit card fraud application. This paper proposes data mining-based detection algorithms.

The first new layer is Communal Detection (CD): the whitelist-oriented approach on a fixed set of attributes. To complement and strengthen CD, the second new layer is Spike Detection (SD): the attribute-oriented approach on a variable-size set of attributes.

The following steps are applied in CD algorithm Multiattribute link, Single-link communal detection, Multiple-links score, Parameter’s value change, Whitelist change.

The following steps are applied in SD algorithm Single-step scaled count, Single-value spike detection, Multiple-values score, SD attributes selection, CD attribute weights change.

**System Architecture**

Credit card Application

Apply



**Admin**

**User**

Receive

**Credit Card Fraud Detection**

Communal Detection

Spike Detection

Detected Result

**Modules**

1. User application
2. Communal Detection
3. Spike Detection
4. Evaluation Measure

**Module Description**

1. **User application**

In this module the GUI is created for credit card application. The user fills the following details, First Name, Last Name, Date of Birth, Street Name and Phone No. The Filled application is send to Admin for verification.

1. **Communal Detection**

In this module the admin use CD algorithm for detecting the credit card fraud. The following steps are applied in CD algorithm Multiattribute link, Single-link communal detection, Multiple-links score, Parameter’s value change, Whitelist change.

1. **Spike Detection**

In this module the admin use SD algorithm for detecting the credit card fraud. The following steps are applied in SD algorithm Single-step scaled count, Single-value spike detection, Multiple-values score, SD attributes selection, CD attribute weights change.

1. **Evaluation Measure**

This module evaluates the performance of crime detection algorithm. In this module we compute precision, recall, f-measure and Roc Curve.

The F-measure curve is recommended over other useful measures for the following reasons. First, for confidentiality reasons, precision-recall curves are not used as they will reveal true positives, false positives, and false negatives. Second, in imbalanced class data, ROC curves, AUC, and accuracy understates false positive percentage because they use true negatives.

**Hardware requirements:**

* Processor : Any Processor above 500 MHz.
* Ram : 1 GB.
* Hard Disk : 10 GB.
* Compact Disk : 650 Mb.
* Input device : Standard Keyboard and Mouse.

**Software requirements:**

* Operating System : Windows Xp.
* Technology : Net Beans 7.3

: Jdk1.6

**Software Description**

Java is a simple and yet powerful object oriented programming language and it is in many respects similar to C++. Java originated at Sun Microsystems, Inc. in 1991. It was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. It was developed to provide a platform-independent programming language.

**Platform independent**

Unlike many other programming languages including C and C++ when Java is compiled, it is not compiled into platform *specific machine*, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.

**Java Virtual Machine**

Java was designed with a concept of ‘write once and run everywhere’. Java Virtual Machine plays the central role in this concept. The JVM is the environment in which Java programs execute. It is software that is implemented on top of real hardware and operating system. When the source code (.java files) is compiled, it is translated into byte codes and then placed into (.class) files. The JVM executes these bytecodes. So Java byte codes can be thought of as the machine language of the JVM. A JVM can either interpret the bytecode one instruction at a time or the bytecode can be compiled further for the real microprocessor using what is called a just-in-time compiler. The JVM must be implemented on a particular platform before compiled programs can run on that platform.

**Object Oriented Programming**

Java is an object oriented programming language it has following features:

* Reusability of Code
* Emphasis on data rather than procedure
* Data is hidden and cannot be accessed by external functions
* Objects can communicate with each other through functions

Object Oriented Programming is a method of implementation in which programs are organized as cooperative collection of objects, each of which represents an instance of a class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

**OOP Concepts**

Abstraction  
Encapsulation  
Inheritance  
Polymorphism

**Abstraction**

Abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.

**Encapsulation**  
Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior ; encapsulation serves to separate the contractual interface of an abstraction and its implementation.

**Inheritance**

Inheritance is the process by which one object acquires the properties of another object.

**Polymorphism**

Polymorphism is the existence of the classes or methods in different forms or single name denoting different  
implementations.

**Java is Distributed**

With extensive set of routines to handle TCP/IP protocols like HTTP and FTP java can open and access the objects across net via URLs.

**Java is Multithreaded**

One of the powerful aspects of the Java language is that it allows multiple threads of execution to run concurrently within the same program A single Java program can have many different threads executing independently and continuously. Multiple Java applets can run on the browser at the same time sharing the CPU time.

**Java is Secure**

Java was designed to allow secure execution of code across network. To make Java secure many of the features of C and C++ were eliminated. Java does not use Pointers. Java programs cannot access arbitrary addresses in memory.

**Garbage collection**

Automatic garbage collection is another great feature of Java with which it prevents inadvertent corruption of memory. Similar to C++, Java has a new operator to allocate memory on the heap for a new object. But it does not use delete operator to free the memory as it is done in C++ to free the memory if the object is no longer needed. It is done automatically with garbage collector.

**NetBeans**

The NetBeans IDE is open source and is written in the Java programming language. It provides the services common to creating desktop applications -- such as window and menu management, settings storage -- and is also the first IDE to fully support JDK 5.0 features. The NetBeans platform and IDE are free for commercial and non-commercial use, and they are supported by Sun Microsystems. It can be downloaded from <http://www.netbeans.org/>

**Features and Tools**

|  |
| --- |
|  |

The NetBeans IDE has many features and tools for each of the Java platforms. Those in the following list are not limited to the Java SE platform but are useful for building, debugging, and deploying applications and applets:

Source Code Editor

* Syntax highlighting for Java, JavaScript, XML, HTML, CSS, JSP, IDL
* Customizable fonts, colors, and keyboard shortcuts
* Live parsing and error marking
* Pop-up Javadoc for quick access to documentation
* Advanced code completion
* Automatic indentation, which is customizable
* Word matching with the same initial prefixes
* Navigation of current class and commonly used features
* Macros and abbreviations
* Goto declaration and Goto class
* Matching brace highlighting
* JumpList allows you to return the cursor to previous modification

**GUI Builder**

* Fully WYSIWYG designer with Test Form feature
* Support for visual and nonvisual forms
* Extensible Component Palette with preinstalled Swing and AWT components
* Component Inspector showing a component's tree and properties
* Automatic one-way code generation, fully customizable
* Support for AWT/Swing layout managers, drag-and-drop layout customization
* Powerful visual editor
* Support for null layout
* In-place editing of text labels of components, such as labels, buttons, and text fields
* JavaBeans support, including installing, using, and customizing properties, events, and customizers
* Visual JavaBean customization -- ability to create forms from any JavaBean classes
* Connecting beans using Connection wizard
* Zoom view ability

**Database Support**

* Database schema browsing to see the tables, views, and stored procedures defined in a database
* Database schema editing using wizards
* Data view to see data stored in tables
* SQL and DDL command execution to help you write and execute more complicated SQL or DDL commands
* Migration of table definitions across databases from different vendors
* Works with databases, such as MySQL, PostgreSQL, Oracle, IBM DB2, Microsoft SQL Server, PointBase, Sybase, Informix, Cloudscape, Derby, and more

The NetBeans IDE also provides full-featured refactoring tools, which allow you to rename and move classes, fields, and methods, as well as change method parameters. In addition, you get a debugger and an Ant-based project system.

**Conclusion**

The main focus of this paper is Resilient Identity Crime Detection; in other words, the real-time search for patterns in a multilayered and principled fashion, to safeguard credit applications at the first stage of the credit life cycle. This paper describes an important domain that has many problems relevant to other data mining research. It has documented the development and evaluation in the data mining layers of defence for a real-time credit application fraud detection system. In doing so, this research produced three concepts (or “force multipliers”) which dramatically increase the detection system’s effectiveness (at the expense of some efficiency). These concepts are resilience (multilayer defence), adaptivity (accounts for changing fraud and legal behavior), and quality data (real-time removal of data errors). These concepts are fundamental to the design, implementation, and evaluation of all fraud detection, adversarial-related detection, and identity crime-related detection systems.

The implementation of CD and SD algorithms is practical because these algorithms are designed for actual use to complement the existing detection system. Nevertheless, there are limitations. The first limitation is effectiveness, as scalability issues, extreme imbalanced class, and time constraints dictated the use of rebalanced data in this paper. The counter-argument is that, in practice, the algorithms can search with a significantly larger moving window, number of link types in the whitelist, and number of attributes. The second limitation is in demonstrating the notion of adaptivity. While in the experiments, CD and SD are updated after every period, it is not a true evaluation as the fraudsters do not get a chance to react and change their strategy in response to CD and SD as would occur if they were deployed in real life (experiments were performed on historical data).

**References**

[1] A. Bifet and R. Kirkby Massive Online Analysis, Technical Manual, Univ. of Waikato, 2009

[2] R. Bolton and D. Hand, “Unsupervised Profiling Methods for Fraud Detection,” Statistical Science, vol. 17, no. 3, pp. 235-255, 2001

[3] P. Brockett, R. Derrig, L. Golden, A. Levine, and M. Alpert, “Fraud Classification Using Principal Component Analysis of RIDITs,” The J. Risk and Insurance, vol. 69, no. 3, pp. 341-371, 2002, doi: 10.1111/1539-6975.00027.

[4] R. Caruana and A. Niculescu-Mizil, “Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria,” Proc. 10th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD ’04), 2004, doi: 10.1145/1014052.1014063.

[5] P. Christen and K. Goiser, “Quality and Complexity Measures for Data Linkage and Deduplication,” Quality Measures in Data Mining, F. Guillet and H. Hamilton, eds., vol. 43, Springer, 2007, doi: 10.1007/978-3-540-44918-8.

[6] C. Cortes, D. Pregibon, and C. Volinsky, “Computational Methods for Dynamic Graphs,” J. Computational and Graphical Statistics, vol. 12, no. 4, pp. 950-970, 2003, doi: 10.1198/1061860032742.

[7] Experian. Experian Detect: Application Fraud Prevention System, Whitepaper, http://www.experian.com/products/pdf/experian\_detect.pdf, 2008.

[8] T. Fawcett, “An Introduction to ROC Analysis,” Pattern Recognition Letters, vol. 27, pp. 861-874, 2006, doi: 10.1016/j.patrec. 2005.10.010

[9] A. Goldenberg, G. Shmueli, R. Caruana, and S. Fienberg, “Early Statistical Detection of Anthrax Outbreaks by Tracking Over-the- Counter Medication Sales,” Proc. Nat’l Academy of Sciences USA (PNAS ’02), vol. 99, no. 8, pp. 5237-5240, 2002

[10] G. Gordon, D. Rebovich, K. Choo, and J. Gordon, “Identity Fraud Trends and Patterns: Building a Data-Based Foundation for Proactive Enforcement,” Center for Identity Management and Information Protection, Utica College, 2007.

[11] D. Hand, “Classifier Technology and the Illusion of Progress,” Statistical Science, vol. 21, no. 1, pp. 1-15, 2006, doi: 10.1214/ 088342306000000060.

[12] B. Head, “Biometrics Gets in the Picture,” Information Age, pp. 10-11, Aug.-Sept. 2006.

[13] L. Hutwagner, W. Thompson, G. Seeman, and T. Treadwell, “The Bioterrorism Preparedness and Response Early Aberration Reporting System (EARS),” J. Urban Health, vol. 80, pp. 89-96, 2006

[14] IDAnalytics, “ID Score-Risk: Gain Greater Visibility into Individual Identity Risk,” Unpublished, 2008.

[15] M. Jackson, A. Baer, I. Painter, and J. Duchin, “A Simulation Study Comparing Aberration Detection Algorithms for Syndromic Surveillance,” BMC Medical Informatics and Decision Making, vol. 7, no. 6, 2007, doi: 10.1186/1472-6947-7-6.

[16] J. Jonas, “Non-Obvious Relationship Awareness (NORA),” Proc. Identity Mashup, 2006.

[17] M. Kantarcioglu, W. Jiang, and B. Malin, “A Privacy-Preserving Framework for Integrating Person-Specific Databases,” Proc. UNESCO Chair in Data Privacy Int’l Conf. Privacy in Statistical Databases (PSD ’08), pp. 298-314, 2008, doi: 10.1007/978-3-540- 87471-3\_25.

[18] J. Kleinberg, “Temporal Dynamics of On-Line Information Streams,” Data Stream Management: Processing High-Speed Data Streams, M. Garofalakis, J. Gehrke, and R. Rastogi, eds., Springer, 2005.

[19] O. Kursun, A. Koufakou, B. Chen, M. Georgiopoulos, K. Reynolds, and R. Eaglin, “A Dictionary-Based Approach to Fast and Accurate Name Matching in Large Law Enforcement Databases,” Proc. IEEE Int’l Conf. Intelligence and Security Informatics (ISI ’06), pp. 72-82, 2006, doi: 10.1007/11760146.

[20] J. Neville, O. Simsek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg, “Using Relational Knowledge Discovery to Prevent Securities Fraud,” Proc. 11th ACM SIGKDD Int’l Conf. Knowledge Discovery in Data Mining (KDD ’05), 2005, doi: 10.1145/ 1081870.1081922.