Situational Awareness Framework for Risk Ranking

by

Rongsha Li

B.E., Tsinghua University (2013)

Submitted to the School of Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Computation for Design and Optimization

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

© Massachusetts Institute of Technology 2015. All rights reserved.

Signature redacted

Author ..............................................................

Signature redacted

School of Engineering
May 18, 2015

Certified by ........
Abel Sanchez
Principal Investigator, Research Scientist, Geospatial Data Center

Signature redacted

Thesis Supervisor

Certified by ........
John R. Williams
Professor of Civil and Environmental Engineering

Signature redacted

CDO Reader

Accepted by ........
Nicolas G. Hadjiconstantinou
Co-Director, Computation for Design and Optimization
Abstract

Today, organizations are generating large volumes of data. However, the challenge of extracting valuable information from the data has been a large and long-standing problem. Here, we address the problem of quantifying risks and detecting fraud in heterogeneous financial big data.

Great financial losses are pressuring institutions to devise innovative solutions for risk and fraud detection. Current approaches in government suffer from issues such as high false positive rates and low adaptability to the continuous evolution of newer fraud.

In this thesis, we propose an open and extensible framework called "Situational Awareness Framework for Risk Ranking" (SAFARI). SAFARI aims to quantify and rank risk with unlabeled, complex data in the financial world. The framework integrates and analyzes different perspectives of financial data, and extends risk scores for decision makers. SAFARI also utilizes machine learning techniques to learn from examined cases to improve the calculation of risks and adapt to the changing behavior of fraudulent activities.

The work includes designing, implementing, testing, extending and evaluating the proposed framework. In the overpayment detection scenario, results show SAFARI can effectively find overpayments with low false positive rates. Furthermore, SAFARI can be extended to assist decision making in a variety of environment thanks to its general applicability.

Thesis Supervisor: Abel Sanchez
Title: Principal Investigator, Research Scientist, Geospatial Data Center
Acknowledgments

Firstly, I would like to thank my thesis advisor, Dr. Abel Sanchez, for introducing me to this interesting project, and for his time, guidance and encouragement. I would also like to thank Professor John R. Williams, for being my thesis reader.

I would like to express my appreciation to Alberto Garcia Robledo, who worked closely with me and gave me insightful advice in many situations.

During my stay at the Comptroller Office at the Commonwealth of Massachusetts, Peter Scavotto, George Anastos and Biruk Gebreamlak introduced the auditing process to me, and made the evaluation of SAFARI possible. Many thanks are due to them.

This work was supported by funding from Accenture. I would also like to thank Oonagh Oshea from Accenture for taking care of the administrative work, including scheduling the bi-weekly meeting.

Beyond this thesis, I am also grateful for my friends and classmates who discussed interesting topics - both academic and non-academic - with me, including Nan Wei, Cong Yan, Zhengdi Shen, Zeyuan Allen-Zhu, Li Jin, Mo Jiang, Zhilei Xu, Jiajun Wu, and numerous others who I cannot name for the list is too long.

Last but not the least, I would like to thank my family for their support and encouragement, and my friends at MIT for making my stay at MIT fun and memorable.
# Contents

## 1 Introduction
1.1 Challenges and Opportunities in Financial Fraud Detection ........................................... 16
1.2 Scoring Scheme for False Positive Reduction ............................................................... 16
1.3 Proposed Framework Architecture ................................................................................... 17
1.4 Implementation and Evaluation ....................................................................................... 18
1.5 Thesis Structure .............................................................................................................. 19

## 2 Design of the Platform Architecture
2.1 Workflow ....................................................................................................................... 21
  2.1.1 Data Access Layer (DAL) .......................................................................................... 22
  2.1.2 Flag Layer (FAL) ..................................................................................................... 22
  2.1.3 Integration Layer (NIL) ............................................................................................ 23
  2.1.4 Ranker Layer (RAL) ................................................................................................. 24
  2.1.5 Situation Comprehension Layer (SCL) ..................................................................... 27
  2.1.6 Visualization Layer (VIL) ......................................................................................... 30
  2.1.7 User Interface ........................................................................................................... 30
2.2 Technical Implementation ................................................................................................ 30
  2.2.1 Software Architecture .............................................................................................. 31
  2.2.2 Components .............................................................................................................. 31

## 3 Scenario Example: Overpayment due to Duplicate Invoices
3.1 Practical Significance ....................................................................................................... 35
3.2 Datasets ........................................................................................................................... 36
3.2.1 Data Facts ......................................................... 36
3.2.2 Selected Relevant Data Fields ............................... 37
3.3 Data Ingestion .......................................................... 39
3.4 Data Enrichment .......................................................... 39
  3.4.1 Geolocation Enrichment ........................................ 39
  3.4.2 Data Statistics Enrichment .................................... 40
3.5 Flag Layer (FAL) ....................................................... 40
3.6 Integration Layer (NIL) ............................................... 42
3.7 Ranker Layer (RAL) ................................................... 43
3.8 Situation Comprehension Layer (SCL) ......................... 45
  3.8.1 Features ........................................................... 45
  3.8.2 Learning Algorithms .......................................... 48
  3.8.3 Theoretical Comparison ....................................... 49
  3.8.4 Practical Comparison .......................................... 49
3.9 Visualization Tools .................................................... 50
  3.9.1 Treemap View .................................................... 50
  3.9.2 Individual RFNet View ....................................... 51
3.10 User Interface (UI) .................................................... 53
3.11 Miscellaneous Details ............................................... 57
  3.11.1 Data Encryption ............................................... 57
  3.11.2 Data Compression .............................................. 58

4 Results and Evaluation of Overpayment Detection ............ 61
  4.1 Evaluation Background ............................................ 61
  4.2 Results .............................................................. 61
  4.3 Comparison of Algorithms in SCL ............................... 62
  4.4 False Positive Rate Reduction ................................. 65
  4.5 Score Distributions ............................................... 66
  4.6 False Positive Examples ......................................... 66
    4.6.1 Department-generated Invoice Numbers .................. 67
5 Contributions of the SAFARI Platform

5.1 Increased Levels of Intelligence ........................................ 71
5.2 Reduced False Positive Rate ........................................... 71
5.3 Flexibility ................................................................. 72
   5.3.1 Use Existing Framework Blocks .................................... 72
   5.3.2 Construct New Components ........................................ 72
5.4 Future Work .............................................................. 73
   5.4.1 Construct Vendor Profiles .......................................... 73
   5.4.2 Other Financial Deceptions ......................................... 73
   5.4.3 Risk Ranking for Decision Makers ................................. 73
List of Figures

1-1 SAFARI workflow .................................................. 17
2-1 Red flags integrated into red flag networks (RFNets) .............. 24
2-2 NIL produces multiple red flag networks (RFNets) ................. 25
2-3 Bayesian belief network ranking: example ranker ................... 26
2-4 Red flag network takes its links' highest risk score as its risk score . 27
2-5 Ranked RFNets .......................................................... 28
2-6 Workflow for Situation Comprehension Layer ...................... 29
2-7 SAFARI software architecture ...................................... 31
3-1 BBN of RAL ............................................................. 43
3-2 SME-informed feature example 1 .................................... 47
3-3 SME-informed feature example 2 .................................... 48
3-4 VIL: treemap view .................................................... 50
3-5 VIL: treemap view with RFNets statistics ........................ 51
3-6 VIL: RFNet view with statistics .................................... 52
3-7 VIL: RFNet view with map .......................................... 53
3-8 Screen shot of User Interface opening page ......................... 53
3-9 Detection tab: select a detector .................................... 54
3-10 Edit selected detector .................................................. 55
3-11 Edit detectors done .................................................... 56
3-12 Run detectors .......................................................... 57
3-13 Check if the detectors are done - partially done .................. 58
3-14 Check if the detectors are done - all done ........................ 59
<table>
<thead>
<tr>
<th></th>
<th>Verification rate enhancement</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>False positive rate reduction</td>
<td></td>
</tr>
</tbody>
</table>

62
65
List of Tables

3.1 Selected data field names and their description .................................. 39
3.2 Conditional probability table at risk score node .............................. 44
3.3 Probability constants ................................................................. 44
3.4 Conditional probability table at vendor match node ...................... 44
3.5 Conditional probability table at VCC_TIN match node .................. 45
3.6 Conditional probability table at VInfo match node ....................... 45
3.7 Conditional probability table at vendor name match node .............. 46
3.8 Conditional probability table at vendor address match node .......... 46

4.1 In-sample test result example 1 .................................................. 63
4.2 In-sample test result example 2 .................................................. 64
4.3 Leave-one-out cross-validation test result example 1 ..................... 64
4.4 Leave-one-out cross-validation test result example 2 ..................... 64
4.5 Risk score distribution of RAL for PRC FY2012 data ..................... 66
4.6 Risk score distribution of SCL for PRC FY2012 data ..................... 66
Chapter 1

Introduction

Today, organizations are faced with overwhelming amount of data. While the data contains valuable information, the challenge of extracting them from large volumes of complex data continues to be a large problem.

Here, we narrow down the problem, and concentrate on quantifying risks from heterogeneous financial data.

There are still many types of risks in financial context. While our vision is to extend our solution to many fields, we are typically interested in the risk of financial fraud as a starting point.

In this thesis, we propose and implement a framework called "Situational Awarness FrAamework for Rlsk ranking" (SAFARI). SAFARI quantifies and ranks risk based on integrating algorithmically raised red flags from different perspectives of the data.

SAFARI is different from the current transactional ACL audit workflow at CTR (Office of the Comptroller) in that it makes use of a variety of detection algorithms, machine learning techniques and visualization tools that exploits different aspects of data. These up-to-date techniques help SAFARI to reduce false positive rates, adapt to the evolution of fraudulent activities, and better assist experts on the identification of fraud on large volumes of complex financial data.

SAFARI is an open and extensible framework that is designed to adapt to new business environment and new techniques.
1.1 Challenges and Opportunities in Financial Fraud Detection

Financial fraud can take a variety of forms in different contexts. Some examples are [13]: bank fraud (e.g. money laundering, credit card and mortgage fraud), insurance fraud (e.g. application, eligibility and claim fraud), and corporate fraud (falsification of corporate financial information and self-dealing by corporate insiders).

From an economic perspective, fraud becomes a worldwide increasingly serious problem. It is estimated that a typical organization loses 5% of revenues each year to fraud, which would be $3.7 trillion in 2013. The median loss caused by the frauds was $145,000, and 22% of the cases involved losses of at least $1 million [14].

Financial losses due to fraud are pressuring institutions to devise solutions for the detection of financial deceptions. Unfortunately, there is a list of critical issues that make fraud detection a rather challenging task, including the large size and complexity of the data, the lack of labeled data, the changing behavior of participants and the high false alarm rate [15].

Current widely used query-based fraud detection approaches, such as ACL, tends to produce a great deal of false positives on large databases with numerous "fraud symptoms" [9]. False positives are a major issue as a false positive requires spending expensive analyst time examining the reported incident only to eventually determine that it reflects benign underlying activity [20].

1.2 Scoring Scheme for False Positive Reduction

False positive rate is a widely-used measurement of the success of fraud departments in financial institutions, and its minimization is one of the objectives of SAFARI.

We propose a scoring scheme in order to quantify the fraudulent risk involved in the incidents, and establish an ordering among suspicious activities.

The idea is to prioritize the examination of incidents that are more likely to be fraudulent. The user can choose a suitable cutoff threshold and examine cases with
high risk scores first.

It is important to make sense of the algorithmically produced risk score. Visualization tools are designed to show how the red flags raised by different detectors contribute to the final score and to facilitate further analysis.

1.3 Proposed Framework Architecture

The framework has 6 layers (corresponds to the 6 Servers) as shown in figure 1-1. They are:

**Data Access Layer (DAL)** that ingests and enriches data. It is responsible for accessing data from heterogeneous data sources, and integrating them into a uniform representation which is suitable for later data processing. It also enriches the ingested data by reaching out to the Internet and bring in additional information (such as geographical information, credit records or even social network information), and by computing data statistics of interests.
Flag Layer (FAL) that examines different aspects of data and creates linked relationships (i.e. raises red flags) between data records. FAL feeds its detectors with appropriate data structures and executes a variety of detection algorithms to look for anomalies from different perspectives. FAL takes advantage of the parallel architecture to enable efficient execution.

Integration Layer (NIL) that integrates the linked relationships generated by FAL to form red flag networks.

Rank Layer (RAL) that assigns risk scores to the red flag networks generated by NIL. RAL uses Bayesian belief network to combine different red flags raised by FAL.

Situation Comprehension Layer (SCL) that uses machine learning techniques to combine customized rules and learn from labeled data when possible.

Visualization Layer (VIL) that visualizes analysis results and presents to SMEs and decision makers. VIL aims to facilitate the cognitive process and allow SMEs to gain further insight into the relationships between alerts.

A Web-based Graphical User Interface (WebGUI) is designed for SMEs to use and interact with SAFARI platform without the need of programming.

1.4 Implementation and Evaluation

The project is conducted with the Quality Assurance Bureau (QAB) in Office of the Comptroller (CTR) at the Commonwealth of Massachusetts (CoM). The State has a large volume of government purchase data from numerous vendors, and the QAB wants to check if there are any duplicated payments - any bills that are paid more than once.

Thus, the proposed platform is implemented and tested on a scenario example: overpayment due to duplicate invoices. The objective is to find payments that are likely to be duplicate by looking at the transaction data and the vendor profile.
Not only governments have such overpayment issue. Large companies that have many upstream suppliers are also very interested in checking if they have made any duplicate payments.

However, finding duplicate invoice number alone would generate a large number of false positives, since people have various legitimate reasons to make multiple payments with a same invoice number. We need to check many more fields, such as the vendor name, the encumbrance, the cost object, etc. to determine whether two payments with the same invoice number are duplicate.

Chapter 3 describes how SAFARI handles this problem in details, and explains the configuration of each component.

According to the evaluation from auditors at QAB, SAFARI can effectively detect duplicate payments with low false positive rates. Chapter 4 discusses the results and evaluation of the overpayment detection.

1.5 Thesis Structure

The structure of this thesis is as follows:

- Chapter 2 introduces the conceptual architecture of the proposed framework, with the functionality and technical implementation of each component.

- Chapter 3 describes the process of using the framework to detect overpayment due to duplicate invoices.

- Chapter 4 discusses the results of the overpayment detection.

- Chapter 5 concludes the contribution of the SAFARI platform.
Chapter 2

Design of the Platform Architecture

2.1 Workflow

The components of SAFARI are organized in 6 layers, as shown in figure 1-1. They are:

Data Access Layer (DAL)
that ingests rich, heterogeneous data and enriches them by reaching out to the Internet and bring in additional information

Flag Layer (FAL)
that examines different aspects of data and creates linked relationships between data records

Integration Layer (NIL)
that integrates all linked relationships to form sophisticated graphs

Rank Layer (RAL)
that ranks risk by probabilistic analysis

Situation Comprehension Layer (SCL)
that uses machine learning techniques to combine customized rules and learn from labeled data (if there are any)
Visualization Layer (VIL)

that visualizes analysis results and presents to decision makers

The WebGUI is designed for Subject Matter Experts (SMEs) to use and interact with SAFARI platform without the need of programming.

2.1.1 Data Access Layer (DAL)

The Data Access Layer (DAL) is responsible for accessing and storing data from heterogeneous data sources. It also integrates the heterogeneous data into a uniform representation which is suitable for the detection algorithms at the Flag Layer.

Data Ingestion

The first step is to ingest data from several databases. Data ingestion component bridges data sources and the framework database.

It supports different data types such as string, float number and date&time.

Data Enrichment

Data enrichment component enriches existing data by reaching out the Internet and bring in addition data.

For example, according to the address database, we may find the corresponding geolocation information (i.e. longitude and latitude); according to vendor profile, we may find its credit ranking or social network information.

Additionally, some data statistics can be computed at this stage for later usage.

2.1.2 Flag Layer (FAL)

The Flag Layer is the key component for link analysis.

Flag layer consists of many detectors to detect anomalies and raise red flags.

Detecting suspicious transactions requires implementing several types of "detectors" to effectively cover a variety of potential risk types. In FAL, we have a bunch of detectors to examine different aspects of data.
Each detector examines certain type of anomalies and creates linked relationships (i.e. raises red flags) between data records. Each detector attaches a tag to the red flags it raises. Two documents can be linked by several red flags with different tags.

FAL feeds different detectors with appropriate data structure. Each detector runs its specific algorithm to look for suspicious activities. FAL uses parallel architecture to enable efficient execution.

As an example, 4 kinds of detectors are implemented:

**N-way exact matching** Matches pairs of documents that have exactly the same values at N different specified data fields.

**Geolocation matching** Matches pairs of documents whose geographical distance is below a given threshold.

**Fuzzy string matching** Matches documents that have a high string similarity in the specified field by exploiting different string fuzzy matching algorithms, including Levenshtein distance [10], Damerau-Levenshtein Distance [2], Jaro Distance [7], Jaro-Winkler Distance [21] and Hamming Distance [5].

**Phonetic string matching** Matches documents that have the same or similar pronunciation with different phonetic encoding system, including NYSIIS (New York State Identification and Intelligence System), American Soundex, Metaphone [16], and Match Rating Codex.

Platform users can specify other types of detectors as needed.

### 2.1.3 Integration Layer (NIL)

NIL creates red flags networks (RFNets) from the red flags raised by detectors in FAL (as shown in figure 2-1).

NIL supports integrating red flags raised by specified detectors (the tag of red flags represents its origin), but we usually use NIL to integrate all red flags raised in FAL and form sophisticated RFNets.
As a result, NIL presents a number of red flag networks to us (as shown in figure 2-2). The next task would be assigning risk scores to those RFNets.

### 2.1.4 Ranker Layer (RAL)

Ranker Layer’s objective is to assign risk scores to the red flag networks generated by NIL.

RAL would firstly assign a risk score to each link between documents by examining the red flags linking the two documents. Two documents can be linked by more than one red flag, but the link between two document would have one risk score.

Then, RAL will find the highest link risk score in a RFNet, and use that as the RFNet risk score. Thus, the risk score for a red flag network is determined by the highest risk score among its edges.
Figure 2-2: NIL produces multiple red flag networks (RFNets)

**Risk Score for Links**

Bayesian Belief Network (BBN) [8][1] is used for red flag fusion on document links.

Figure 2-3 shows an example of how BBN integrates the red flags raised by FAL, and calculate a ranking score. Here, the link between record A and record B has 3 red flags, raised by Fuzzy Matching, Phonetic Matching, and Geolocation Matching, respectively.

Suppose the BBN constructed by the Example Ranker is as shown in figure 2-3. The probabilities that node a=1, b=1, c=1, d=1 are 1, 1, 1, 0, respectively. Suppose the BBN Conditional Probability Table (CPT) at node e (NameMatch) has 2 rows as shown in table 2.1.4:

<table>
<thead>
<tr>
<th>NameFuzzyMatch</th>
<th>NamePhonMatch</th>
<th>NameMatch</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

That means that if "NameFuzzyMatch" = 1 AND "NamePhonMatch" = 1, then
"NameMatch" = 1 with probability 1. Thus, Node e is 1 with probability 1.

Similarly, suppose the CPT at node f has 2 rows as shown in table 2.1.4:

<table>
<thead>
<tr>
<th>AddressGeoMatch</th>
<th>AddressNwayMatch</th>
<th>AddressMatch</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
</tbody>
</table>

That means if "AddressGeoMatch" = 1 and "AddressNwayMatch" = 0, then "AddressMatch" = 1 with probability 0.8, and "AddressMatch" = 0 with probability 0.2.

Finally, suppose the CPT at node r (which produces "Vendor Match" risk score) has 4 rows as shown in table 2.1.4. Please note that the CPT do not have to be what we supposed in table 2.1.4 to produce a "VendorMatch" = 0.95.

<table>
<thead>
<tr>
<th>NameMatch</th>
<th>AddressMatch</th>
<th>VendorMatch</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
</tr>
</tbody>
</table>

That means if "NameMatch" = 1 and "AddressMatch" = 1, then "VendorMatch" = 1 with probability 1; one the other hand, if "NameMatch" = 1 but "AddressMatch" = 0,then "VendorMatch" can still be 1 with probability 0.75.

In this way, RAL combined the possibly multiple red flags between two records to produce 1 risk score: 0.95.
Risk Score for Red Flag Networks

After calculating risk scores for each link, RAL simply takes the highest risk score within a red flag network as its risk score, as shown in figure 2-4.

In this way, we can find the key links within piles of red flag networks.

![Figure 2-4: Red flag network takes its links' highest risk score as its risk score](image)

Risk Ranking for RFNets

After calculating the risk scores for each RFNets, we can pinpoint the key RFNet and red flags and extend them to SMEs for further inspection, as figure 2-5 shows.

2.1.5 Situation Comprehension Layer (SCL)

Situation Comprehension Layer adjusts risk scores given by RAL, by utilizing a few labeled data.

Motivation

RAL ranks risk without any labeled data. It solves the risk ranking problem that we initially faced with, when no labeled data is available. However, as we get our first patch of high-risk cases, we have some Subject Matter Experts (SMEs) to research on those cases, and we will get the corresponding labels - whether a case is true positive or false positive - according to their feedback.
Situation Comprehension Layer is used to preserve those newly acquired information and exploit it to improve future risk ranking.

By adding SCL, SAFARI can help SMEs to focus on the cases similar to previously confirmed risks (i.e. true positives), and can prevent SMEs from examining a large number of cases similar to known false positive.

This largely enhances SAFARI’s ability to reduce false positives.

**Workflow**

Implementing the Situation Comprehension Layer includes the following procedures:

- Construct a database to store labeled data.

- Compute "features" for both labeled and unlabeled data. The "features" can be self-generated by SAFARI, but in order to get better performance, SMEs
input is essential. So we have both auto-generated features and SME informed features.

- Adopt machine learning techniques to compute modified risk scores. This can also be viewed as using machine learning techniques to find a "good" way to combine multiple SME informed rules.

SME feedback plays an important role in this process. Since only a small number of labeled data is available, we should take extra care of the machine learning procedure to avoid overfitting.

**Features**

We have both auto-generated features and SME informed features.
Auto-generated features can be easily obtained. One possible implementation can be boolean values comparing whether two fields have the same value. Moreover, one can define more features by doing AND/OR operations on these boolean values.

SME-informed features are features specified by SMEs. SME-informed features are highly flexible, and usually have much better prediction abilities. However, they are harder to get, and requires intensive expert knowledge.

Learning Algorithms

Machine learning techniques are used to combine those features.

Typical supervised learning algorithms, such as Regression Models, Support Vector Machines, Neural Networks, Nearest Neighbors, Decision Trees, and Ensemble methods can be used here.

2.1.6 Visualization Layer (VIL)

The Visualization Layer provides an interactive web-based front-end visualization tool. The objective of VIL is to facilitate the cognitive process and allow SMEs to gain further insight into the relationships among alerts.

In addition, the visualization tool offers visual summaries of the red flags, ranked RFNets and other final results.

2.1.7 User Interface

User Interface is designed for SMEs to use and interact with the SAFARI platform without the need of programming. It has interface with each layer of SAFARI, and offers high flexibility and rich functionality.

2.2 Technical Implementation

The platform is mainly implemented in Python, except the visualization front-end which is written in html and Javascript.
2.2.1 Software Architecture

The layers are decoupled by Representational State Transfer (REST) servers. The layers exchange JSON documents to communicate. MongoDB is used for storing data at each REST server.

2.2.2 Components

Database

MongoDB is used as the database for each layer. MongoDB is an open-source NoSQL document database. Its documents are JSON style, with dynamic schemes. MongoDB queries are also document-based, with its own grammar. It supports flexible aggregation and map-reduce data processing.

Web Application Server

Flask is a lightweight web application framework for Python. It is a microframework that keeps the core simple, but is still extensible. Flask supports
REST style.

Data Structure and Data Analysis Tool

Pandas[11] is a library that provides high-performance, easy-to-use data structures and data analysis tools. It is used in DAL.

Task Queue

Celery[19] is an asynchronous task queue/job queue based on distributed message passing. Tasks are executed concurrently on a single or more worker servers. It helps servers (FAL, NIL, RAL and SCL) to execute its tasks in parallel.

RFNet Integrator and Formatter

igraph[6] is a collection of network analysis tools with the emphasis on efficiency, portability and ease of use. It provides algorithms for measuring structural properties, node centrality, K-decomposition and community detection. It also has algorithms for generating 2D/3D layouts, such as Fruchterman-Reingold, Kamada-Kawai, Distributed Recursive Layout, etc. Igraph is used in the integrator in NIL and the Formatter in VIL.

Bayesian Belief Network

EBay’s Bayesian Belief Network[17] project is a Pythonic Bayesian Belief Network package that supports discrete and continuous variables and has different inference engines, such as junction tree and sum product.

It is used in Rank Layer to fuse red flags and generate risk scores.

Machine Learning Algorithms

Scikit-learn [18] is a widely-used machine learning package in Python for data mining and data analysis. It has a number of popular machine learning algorithms. For supervised learning, it has Regression Models, Support Vector Machines, Nearest Neighbors, Decision Trees, etc. For unsupervised learning, it has Gaussian Mixture Models, Clustering, Decomposing signals in components, unsupervised Neural Network Models, etc. It also supports some semi-supervised learning, feature selection, etc.
The algorithms in this package is used in Situation Comprehension Layer.

**Visualization Tools and User Interface**

**JavaScript InfoVis Toolkit** provides tools for creating interactive data visualizations for the web. It is based on the HTML5 canvas, and supports graph, radial and hierarchical network visualizations. It has treemap, stacked, area, bar and pie charts.

**jQuery EasyUI** is a collection of user-interface plugin based on HTML5 and jQuery. It has essential functionality for building interactive javascript applications. It supports datagrid, treegrid, panel, and many more for building cross-browser web pages.
Chapter 3

Scenario Example: Overpayment due to Duplicate Invoices

The objective of the duplicate invoice scenario is to find payments that are likely to be duplicate by looking at the transaction data and the vendor data. In this chapter, we show how to use the proposed framework to identify overpayment due to duplicate invoices.

3.1 Practical Significance

The project is conducted with the Quality Assurance Bureau (QAB) in Office of the Comptroller (CTR) at the Commonwealth of Massachusetts (CoM). The State has a large volume of government purchase data from numerous vendors, and the QAB wants to check if there are any duplicate payments - any bills that are paid more than once.

Not only governments have such overpayment problem. Large companies that have many upstream suppliers are also very interested in checking if they have made any duplicate payments.

However, to answer of whether duplicate payments have been made, finding duplicate invoice number itself is not enough. People have various legitimate reasons to make multiple payments with a same invoice number: sometimes they need to
split a payment among multiple encumbrances or multiple appropriation, so they use the same invoice number to create payment documents with different encumbrance or appropriation; sometimes the payment is a government fund for municipal governments or departments, and the payment does not get invoices in return, so the processing department generated the invoice numbers themselves, which may be the same throughout all similar payments; sometimes the system does not support more than 99 accounting lines, so the department has to create multiple documents.

The State has over 150 departments, and each department has its own rule to process payments. Coding a specific rule for each department is not realistic, so we are focusing on universal solutions.

### 3.2 Datasets

The payment data we use in this example is the **Commodity-based CoM Payment Document (PRC)** data for 2012 fiscal year.

We also use the **vendor** database, which includes the vendor information.

#### 3.2.1 Data Facts

The PRC data contains transaction-related information, such as the service date, payment amount, payment date, invoice-receiving date, invoice number, encumbrance, appropriation, etc.

The vendor data contains vendor information, such as Taxpayer Identification Number (TIN), Vendor/Customer Code (VCC), legal name, address, etc.

A data unit is called a piece of "record". Each record has both transaction and vendor information.

Each accounting line forms a "record". That means, if a purchase includes 3 different items and has 3 accounting lines, it will form 3 records, though they share the same invoice and are paid altogether in one paycheck.

The purchases in PRC have their corresponding encumbrances. The vendors in PRC data are companies (rather than individuals). Although PRC has "commodity-
"based" in its name, not all purchases involves a physical commodity settlement. Telephone services (whose vendor may be Verizon) or government fund distribution (whose vendor may be public libraries) are also included in PRC dataset.

1 year PRC data is a little more than 1 GB, when stored in .del(similar to .csv) format.

### 3.2.2 Selected Relevant Data Fields

Table 3.1 summarizes some relevant data fields that will be mentioned later. Some field names that are self-explaining (such as "state","city","") are not listed here. Please note that not all fields that are used in this example are included, for both confidentiality and space limit reasons.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appropriation</td>
<td>Appropriation number.</td>
</tr>
<tr>
<td>check_eft_no</td>
<td>Check or Electronic Funds Transfer number.</td>
</tr>
<tr>
<td>encumbrance_doc_identifier</td>
<td>Encumbrance Doc ID.</td>
</tr>
<tr>
<td>fiscal_period</td>
<td>Fiscal period (FP). There are 13 fiscal periods in a fiscal year. July is the 1st FP, June is the 12th FP, etc. The &quot;13th&quot; fiscal period is used to make adjustments after the 12th FP.</td>
</tr>
<tr>
<td>legal_name</td>
<td>Vendor legal name. Such as Home Depot USA Inc, Massachusetts General Hospital, Town of Brookline, etc.</td>
</tr>
<tr>
<td>object_class</td>
<td>&quot;Object class&quot; describes the payment category. For example, object class &quot;AA&quot; stands for &quot;State Employee Compensation&quot;, &quot;EE&quot; stands for &quot;Administrative Expenses&quot;, &quot;JJ&quot; stands for &quot;Programmatic Operational Services&quot;, &quot;PP&quot; stands for &quot;Grants and Subsidies&quot;, etc.</td>
</tr>
<tr>
<td>object</td>
<td>&quot;Object&quot; describes a more detailed category of the payment. For example, &quot;E01&quot; stands for &quot;Office and Administrative Supplies&quot;, &quot;E22&quot; stands for &quot;Temporary Use of Space, Conferences and Conference Incidentals&quot;, &quot;JJ1&quot; stands for &quot;Legal Support Services&quot;, &quot;JJ2&quot; stands for &quot;Auxiliary Services&quot;, etc. Objects starting with &quot;E&quot; (including &quot;EE&quot;) belongs to object class &quot;EE&quot;, and Objects starts with &quot;J&quot; (including &quot;JJ&quot;) belongs to object class &quot;JJ&quot;, etc.</td>
</tr>
<tr>
<td>pymt_actg_line_amount</td>
<td>Payment amount for an accounting line.</td>
</tr>
<tr>
<td>pymt_doc_actg_line_no</td>
<td>Accounting line number (AL).1 AL corresponding to 1 record in SAFARI.</td>
</tr>
<tr>
<td>pymt_doc_comm_line_no</td>
<td>Commodity line number (CL).</td>
</tr>
<tr>
<td>pymt_doc_department_code</td>
<td>Department code. For example, &quot;EHS&quot; stands for &quot;Executive Office of Health and Human Services&quot;.</td>
</tr>
<tr>
<td>pymt_doc_identifier</td>
<td>Doc ID. This is an unique ID for each payment record. When a payment has multiple accounting lines or commodity lines, they share the same Doc ID.</td>
</tr>
<tr>
<td>pymt_service_from_date</td>
<td>Service from date (SVC from)</td>
</tr>
<tr>
<td>pymt_service_to_date</td>
<td>Service to date (SVC to)</td>
</tr>
</tbody>
</table>
### 3.3 Data Ingestion

We use the Data Access Layer to ingest the .del data. To specify the data fields and corresponding data type, a JSON file is sent to DAL server.

As stated in section 2.2.2, the ingested data are stored in MongoDB. Each entry in MongoDB corresponds with a "record" (similar to a "row" in a SQL database).

In our implementation example, there are 46 fields ingested.

### 3.4 Data Enrichment

#### 3.4.1 Geolocation Enrichment

Geo server is used to obtain an address’ geolocation information (i.e. longitude and latitude) and add them into the corresponding record.

Geospatial data is taken from US Census Bureau’s TIGER/Line data set for Massachusetts.
3.4.2 Data Statistics Enrichment

Besides taking in more data, we may want to calculate some data statistics in the enrichment stage as well.

In this scenario example, for every record \( R \), we calculate how many documents (including itself) have the same invoice number with \( R \). This quantity is saved as "inv_no_occurrence". This field is quite useful in the Situation Comprehension Layer. Please note that we count the number of "documents", not "records", that share the same invoice number with \( R \).

3.5 Flag Layer (FAL)

For each detector, we would specify 3 properties:

a) **Grouping** (within what range)

b) **Matching** (detect what kind of anomaly)

c) **Mismatching** (except what conditions)

For example, suppose we want to use fuzzy matching on vendor legal name. we may:

a) specify "grouping" as : "\[
\text{'state', 'city', 'zip_code', 'fiscal_year', 'pymt_doc_department_code', 'fiscal_period'}\]", meaning: find fuzzy matching within the records whose "state", "city", "zip_code", "fiscal_year", "pymt_doc_department_code" and "fiscal_period" are the same;

b) specify "matching" as : "similarity = safari_string_similarity ( doc_a[\'legal_name\'], doc_b[\'legal_name\'], metric = \'levenshtein_distance\', normalized = True ); similarity \geq 0.9", meaning: raise a red flag for each pair of records where the vendor legal names have a Levenshtein string similarity greater than or equal to 90%;

c) specify "mismatching" as : "( doc_a[\'pymt_doc_identifier\'] == doc_b[\'pymt_doc_identifier\'] ) or ( doc_a[\'check_eft_no\'] == doc_b[\'check_eft_no\'] ) or ( 
doc_a['vendor_customer_code'] == doc_b['vendor_customer_code'] and doc_a['tin'] == doc_b['tin']

1 meaning: no pairs of records with the same "Doc ID" or same "check EFT number", or (same VCC and same TIN) will be matched.

If two records have the same "Doc ID" or the same "check EFT number", they should be one payment. It is natural that they meet the criteria specified in "matching". However, we do not want to raise red flags between the two records, since they are one same payment, not duplicate payment.

Because we want to use fuzzy matching detector to find vendors with similar legal names, but not the same vendor, we do not raise red flags between records with the same VCC and same TIN (which means they are the same vendor).

As mentioned in 2.1.2, 4 types of detectors are implemented.

In this scenario example, the "grouping" property for most detectors are "[
'state', 'city', 'zip_code', 'fiscal_year', 'pymt_doc_department_code', 'fiscal_period'
]", except N-way exact matching detectors, whose "grouping" property is the fields they matches.

The "mismatching" property for most detectors are 
( doc_a['pymt_doc_identifier'] == doc_b['pymt_doc_identifier'] ) or ( doc_a['check_eft_no'] == doc_b['check_eft_no'] ) or ( doc_a['vendor_customer_code'] == doc_b['vendor_customer_code'] and doc_a['tin'] == doc_b['tin'] )

except that the "mismatching" property for N-way exact matching b) is "doc_a['pymt_doc_identifier'] == doc_b['pymt_doc_identifier']" or ( doc_a['check_eft_no'] == doc_b['check_eft_no'] )

The "matching" property for the 4 types of detectors are as follows. Each of these detectors attaches a tag to the red flags it raised, and we name the detectors by its tag as well.

N-way exact matching

a) on address (including "country","state","city","address line"), "department", "FY" and "FP". (tag: nway_address_match)

1An equivalent mismatching expression is: 
{ "$or" : [ { "a.pymt_doc_identifier" : "b.pymt_doc_identifier" }, { "a.check_eft_no" : "b.check_eft_no" }, { "a.vendor_customer_code" : "b.vendor_customer_code", "a.tin":"b.tin" } ] }. This expression shows more conspicuous relationship with MongoDB query language.
b) on "invoice number", "payment amount", "SVC from", "SVC to", "department", "FY" and "FP". (tag: nway_invoice_match)

**Geolocation matching** on geolocation. Red flag is raised where two records have the same geographical location. (tag: geolocation_address_match)

**Fuzzy string matching** on "vendor legal name". Red flag is raised where vendor legal names have a Levenshtein string similarity equal or greater than 90%. (tag: fuzzy_lname_match)

**Phonetic string matching** on "vendor legal name". Red flag is raised where every word of the vendor legal names have the same NYSIIS phonetic encoding. (tag: phonetic_lname_match)

As one may have noticed, these input formats are closely related to MongoDB query. In fact, these queries take advantage of the efficient implementation of MongoDB aggregation method.

### 3.6 Integration Layer (NIL)

NIL expects a JSON file with a "query" and a "tag" field. The "query" field specifies the red flags to integrate, and the "tag" field specifies the tag to attach to the integrated red flag networks.

Usually we want the integrator to integrate all the red flags. Suppose we tag the resulting red flag networks "rfnet_dupinvoice". Then the JSON file in this example would be:

```json
{
    'query': {'_tags': {'$in': ['nway_address_match', 'nway_invoice_match', 'geolocation_address_match', 'fuzzy_lname_match', 'phonetic_lname_match']}},
    'tags': ['rfnet_dupinvoice']
}
```
3.7 Ranker Layer (RAL)

The Ranker Layer uses Bayesian Belief Networks (BBN) to compute and combine the probability for pairs of payments of being duplicate based on the Red Flags raised by detectors.

RAL is typically used when there is no or too few labeled data.

Figure 3-1: BBN of RAL

Figure 3-1 shows the BBN structure in RAL for this duplicate invoices scenario example. Each node represents a variable, and the leaf node variables are evidence variables provided by the detectors (such as phonetic name match, geolocation match) or by comparing specific data fields (such as TIN match, invoice match).

The BBN Conditional Probability Table (CPT) at the risk score node (which generates the risk score) is shown in table 3.2.

The value of probability constants are set as table 3.3.

Table 3.2 shows that if both invoice and vendor matches, then the risk score is the highest. It is medium risk if only the invoice matches, and it is low risk if only the vendor matches.
<table>
<thead>
<tr>
<th>Risk Score</th>
<th>Invoice Match</th>
<th>Vendor Match</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>MAX</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>MAX</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>LOW</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>MAX</td>
</tr>
</tbody>
</table>

Table 3.2: Conditional probability table at risk score node

<table>
<thead>
<tr>
<th>Constant</th>
<th>MAX</th>
<th>HIGH</th>
<th>MEDIUM</th>
<th>LOW</th>
<th>MIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0.999</td>
<td>0.75</td>
<td>0.5</td>
<td>0.25</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Table 3.3: Probability constants

The CPT at vendor match node is shown in table 3.4. If both VCC_TIN and VInfo matches, then the vendor match has the highest probability. VCC_TIN Match has a higher weight in determining whether the vendor matches: if only VCC_TIN matches, the vendor has high probability to match, while if only VInfo matches, the vendor only has medium probability to match.

<table>
<thead>
<tr>
<th>Vendor Match</th>
<th>VCC_TIN Match</th>
<th>VInfo Match</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>MAX</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>LOW</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>HIGH</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>MAX</td>
</tr>
</tbody>
</table>

Table 3.4: Conditional probability table at vendor match node

The CPT at VCC_TIN match, VInfo match, vendor name match, vendor address match nodes are shown in tables 3.5, 3.6, 3.7, and 3.8, respectively.
### Table 3.5: Conditional probability table at VCC_TIN match node

<table>
<thead>
<tr>
<th>VCC_TIN Match</th>
<th>VCC Match</th>
<th>TIN Match</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>MAX</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>MAX</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>MAX</td>
</tr>
</tbody>
</table>

### Table 3.6: Conditional probability table at VInfo match node

<table>
<thead>
<tr>
<th>VInfo Match</th>
<th>Vendor Name Match</th>
<th>Vendor Address Match</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>MAX</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>MAX</td>
</tr>
</tbody>
</table>

### 3.8 Situation Comprehension Layer (SCL)

Situation Comprehension Layer adjusts the ranking scores computed by the Ranker Layer.

It helps the framework to learn from labeled data when applicable, and use machine learning techniques to optimally combine the features. Please note that SME-informed features are essentially rules specified by SMEs. SMEs may have already identified those rules, but SCL helps them to find a good way to combine those individual rules.

### 3.8.1 Features

As stated in 2.1.5, there are both auto-generated features and SME-informed features.
<table>
<thead>
<tr>
<th>Vendor Name Match</th>
<th>Phonetic Name Match</th>
<th>Fuzzy Name Match</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>MAX</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>MAX</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>MAX</td>
</tr>
</tbody>
</table>

Table 3.7: Conditional probability table at vendor name match node

<table>
<thead>
<tr>
<th>Vendor Address Match</th>
<th>Nway Address Match</th>
<th>Geolocation Match</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>MAX</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>MIN</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>MIN</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>MAX</td>
</tr>
</tbody>
</table>

Table 3.8: Conditional probability table at vendor address match node

**Auto-generated features**

In this scenario example, we simply use boolean values indicating if the pair has the same value in specific fields as auto-generated features.

The fields compared in this scenario are: SVC from date, SVC to date, payment amount, appropriation, created by, created date, encumbrance number, invoice date, object, pymt unit, TIN number. "Occurrence" (field name: inv_no_occurrence, the number of different Doc IDs that also have this invoice number) is also included in auto-generated features.

The auto-generated features tends to be easy to get, and is more portable from scenario to scenario.

However, it is not easy for them to describe the dependent-relationships among fields. For example, according to SMEs, if two records have same vendor & same encumbrance & different appropriation, it is 6 (unit) LESS risky, whereas if two records still have same vendor & same encumbrance, but have same appropriation, it is only 1
(unit) LESS risky; however, if two records have the same vendor but different encumbrance, it is 6 (unit) MORE risky, regardless of whether the same appropriation or not. Auto-generated features are not good at extracting such dependent relationships especially when only a small number of labeled data is available.

**SME-informed features**

SME-informed features are complicated rules specified by SMEs. They are highly-flexible, and have better prediction ability, though they are harder to get.

In this scenario example, SME-informed features require the two records connected by the evaluating link to have: DIFFERENT Doc ID, SAME invoice number, SAME payment amount, SAME service to date. Otherwise, the SME-informed features are set to 0.

Before computing SME-informed features, a default value and a unit value is set. Suppose we set the default value to 60 and the unit value to 5.

![Figure 3-2: SME-informed feature example 1](image)

Suppose the example in figure 3-2 represents a feature named $x_{SME1}$. Figure 3-2 means:

- if the two records have different service from date (remember that the records also have the same service to date), created by same user, and have the same invoice date, then $x_{SME1}$ for the link connecting those two records is 60 (default) - 1 * 5 (unit) = 55;
• if they still have different service from date, created by same user, but they have different invoice date, then its $x_{SME1}$ is $60 + 3 \times 5 = 75$

• if they have different SVC from date, created by different users, and have the same invoice date, then its $x_{SME1}$ is $60 + 3 \times 5 = 75$

• if they have different SVC from date, created by different users, and have different invoice date, then its $x_{SME1}$ is $60 + 1 \times 5 = 65$

Another SME-informed feature example is shown in figure 3-3.

- same encumbrance: -1 less risky
- diff encumbrance: -6 less risky
- diff encumbrance: +6 more risky
- same encumbrance: +4 more risky
- diff encumbrance: +6 more risky

Figure 3-3: SME-informed feature example 2

Four SME-informed features are used in this scenario example.

### 3.8.2 Learning Algorithms

Since SCL’s object is to learn from a few labeled data, only supervised learning and semi-supervised learning are considered here.

Semi-supervised learning seems to be a perfect choice at first glance, since it focuses on learning tasks with a small amount of labeled data with a large amount of unlabeled data, which is exactly the problem we are faced with. However, typical semi-supervised learning algorithms has strong assumptions on data distribution that are not satisfied in our case, since true positives are not necessarily follow certain distributions.
So we focused on supervised learning algorithms. In this scenario example, Support Vector Classifier (SVC), Random Forest (RF) and Support Vector Regression (SVR) are compared.

### 3.8.3 Theoretical Comparison

Support Vector Machine (including SVC and SVR) is a widely-used method that can be used in small sample scenario. It is easier to train (since it does not have too many parameters).

SVM may not be the best algorithm for specific task, but it usually works reasonably well in most tasks.

Random Forest (RF) is a widely-used ensemble learning method (that works well in many tasks). It has many degrees of freedom. However, it tends to over-fit or have randomness on small data sets.

### 3.8.4 Practical Comparison

A much more detailed comparison is given in Section 4.3.

Both in-sample test and leave-one-out cross-validation are used in our practical comparison.

Four parameters are compared: true positive case count, false positive case count, false negative case count, and the threshold that separates test positive outcome and the test negative outcome.

The Comparison Conclusions are:

- Support Vector Regression performs slightly better than Support Vector Classifier.
- Random Forest has some randomness. The thresholds in two evaluation methods differs less than SVR or SVC.
- Thus, we use the score produced by Support Vector Regression as primary ranking score, and use the score generated by Random Forest as a
3.9 Visualization Tools

Visualization Layer helps SMEs to better understand the inspected data by providing various tools to unveil relationships and gain insights.

3.9.1 Treemap View

Figure 3-4: VIL: treemap view

Figure 3-4 is a Treemap view of all the detected RFNets. The redder the tiles are, the higher risk the corresponding RFNets have; the greener the tiles are, the lower risk the RFNets have. The tiles are sorted in a way that the tile at upper left corner has the highest risk score, and the tile at lower right corner has the lowest risk score.

The panel on the right shows individual tile's statistics, including the number of nodes, links and its tags. There is also an "impact factor" that can be set to adjust the raw risk scores to emphasize RFNets with certain features of interest. Specifically, \( \text{adjustedriskscore} = \text{rawriskscore} \times (1 + \text{impactfactor}) \).
For example, "impact factor" can be set so that a link with higher pymt_actg_line_amount has a higher "impact factor", if we want to emphasize the payments with larger amount. In the example in figure 3-4, the impact factors are set to 0.

The tab "Stats" of the panel on the right shows statistics for all RFNets, as figure 3-5 shows. The statistics includes the number of RFNets, links and nodes in all RFNets, and the number & ratio of RFNets with high, medium and low risk scores.

One may notice that the tiles in figure 3-4 and figure 3-5 have different risk scores. In fact, they are generated by RAL and SCL, respectively. The detailed comparison can be found in section 4.5.

### 3.9.2 Individual RFNet View

By clicking a tile in the Treemap View, the user would see the detailed information of the selected RFNet.

Figure 3-6 shows the **Individual RFNet View** of a Red Flag Network. The center panel shows the network.

The left panel controls the display of the network. In figure 3-6, Node size factor is set to 3, Node style is set to "Ranking", which means the size of the node is determined by its risk score. The link style is also set to "Ranking", meaning the
thickness of the link is also determined by its risk score. (Please note that each node and each link also has its own risk score.) The value in pymt_doc_identifier field is set as node label.

The right panel displays the distribution of values within each field. For example, in figure 3-6, since the two payment documents have the same object_class, UU, so 100% of the documents in this RFNet has object_class UU; since they have different vendor_customer_code, 50% of the documents has VC6000162318, and 50% has VC6000227932.

After clicking on a link, the bottom panel displays the detailed information of the documents (nodes) linked by the selected link.

Figure 3-7 shows the individual RFNet View of another RFNet.

The node and link display are set to default, so the size of nodes and the thickness of links are not determined by their risk scores.

The map on the right panel shows the location of the two vendors associated with the RFNet. From the bottom panel, we can find that the two vendors are from Westfield, MA and South Easton, MA.
3.10 User Interface (UI)

The User Interface is designed for SMEs to interact with the SAFARI platform.
Figure 3-8 shows the User Interface opening page. It has 6 tabs: Ingestion, Enriching, Detection, Integration, Ranking, and Predicting, which correspond to Data Ingestion, Data Enrichment, Flag Layer, Integration Layer, Ranker Layer, and Situation Comprehension Layer, respectively.

At each stage, the user can choose which module to use, and configure the parameters as desired.

Let us take the Detection Layer as an example.

![Detection tab: select a detector](image)

As figure 3-9 shows, after clicking the "Detection" tab, the user can select the type of detectors he would like to use via the ComboBox. He can choose as many detectors as he likes.

Suppose he chooses 5 detectors, with 2 "nway_exact_matching" detectors, and 1 for each other type of detectors.

He can then configure the detector parameters by selecting the corresponding detector, and click "Edit".
In the popped-out dialog, he can type in the desired parameters for the selected detector, as shown in figure 3-10.

Figure 3-11 shows the UI when the editing is done. The console displays the last confirmed detector information. In this example, detector #5 has just confirmed editing.

Each detector has an associated tag, which will be referred to in the integration and the ranking stage. It is allowed to have more than 1 tag.

Since all detectors are ready, the user clicks the "Run detectors" button to run those configured detectors, as shown in figure 3-12.

The Flag Layer Server receives the commands, and is busy raising red flags when certain types of anomalies are detected.

The detecting tasks are executed asynchronously and in parallel, so each task is assigned a "Task ID" by the server. These "Task ID"s are used as references when querying the task status.
Select an option to configure the enrichers, detectors, integrators or rankers.

In figure 3-12, we can see that the "Status" column changed to "OK" (which means the server successfully received the corresponding task), and each detector is assigned a unique "Task ID".

Those tasks would take a while to execute. Of course, the actual execution time largely depends on the detector parameters.

Although the detecting tasks are executed in parallel and they do not have dependencies on each other, all detection tasks have to be done before the integration task can start.

Thus, after running the detectors, we need to check if the tasks are done by clicking the "Check detectors" button.

Figure 3-13 shows a possible outcome, where the two nway_exact_matching detectors are done (whose statuses are "SUCCESS"), but the three other detectors are still running (whose statuses are "PENDING").

After clicking the "Check detectors" button, the UI automatically checks if the
pending tasks are done every 1 second, and lasts for 30 seconds or until all tasks are done. Figure 3-14 is a screen shot when all the detectors are done. If a certain detecting task fails, its corresponding status will be "FAILURE".

After all the tasks are "SUCCESS", the user can proceed to "Integration" tab for further processing.

3.11 Miscellaneous Details

3.11.1 Data Encryption

Since our data set includes TIN number, which is sensitive data, we need to encrypt the data when storing them.

We set up an encrypted folder to store both the .del file and the MongoDB database file.
3.11.2 Data Compression

TokuMX is a high-performance distribution of MongoDB. It uses the same language as MongoDB, but it uses a different format to store the data. It is especially efficient in handling highly repetitive data and reducing the database size.

TokuMX can be used as a substitution of the standard MongoDB in the implementation.
Figure 3-14: Check if the detectors are done - all done
Chapter 4

Results and Evaluation of Overpayment Detection

4.1 Evaluation Background

As stated before, this project is cooperated with the Quality Assurance Bureau (QAB) in Office of the Comptroller at the Commonwealth of Massachusetts (CoM).

The evaluation process is: we firstly report some cases as suspicious (positive) cases, and the Subject Matter Experts examine those cases by contacting the involved departments. In total, we reported 4 patch of cases, in August 2014, October 2014, December 2014, January 2015, respectively.

4.2 Results

Figure 4-1 shows the verification rate enhancement in the development process of SAFARI.

In August 2014, PRC FY2012 data was examined by the platform, and 29 cases were reported. 7 out of the 29 cases were found to be true positives.

In October 2014, we added more fields into the analysis. This time, 33 additional cases were reported. They also came from PRC FY2012 data. 9 out of the 33 cases were true positives.
### Table 4-1: Verification rate enhancement

<table>
<thead>
<tr>
<th>Timeframe</th>
<th>Data</th>
<th># explored</th>
<th># hits</th>
<th>Verification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>August</td>
<td>PRC 2012</td>
<td>29</td>
<td>7</td>
<td>24%</td>
</tr>
<tr>
<td>October</td>
<td>PRC 2012</td>
<td>33</td>
<td>9</td>
<td>27%</td>
</tr>
<tr>
<td>December</td>
<td>PRC 2012</td>
<td>7</td>
<td>5</td>
<td>71%</td>
</tr>
<tr>
<td>January</td>
<td>PRC 2014</td>
<td>13</td>
<td>11</td>
<td>85%</td>
</tr>
</tbody>
</table>

Figure 4-1: Verification rate enhancement

In November 2014, Situation Comprehension Layer was implemented. All August and October cases were used as training data.

In December 2014, 7 additional cases were reported. 5 of them were true positives. They were from PRC FY2012 data.

In January 2015, 13 cases were reported. Note: this time, the cases came from PRC FY2014 data.

Please note that, from August to December, the same set of PRC 2014 data is analyzed, and additional cases are reported each time. The previously reported cases are still caught, but they were of course not reported.

### 4.3 Comparison of Algorithms in SCL

#### Training Data Set

We have 96 pair-wise cases in total, with 17 positive cases. (17.7% positive rate).

Please note that although in August and October 2014, we only reported $29 + 33 = 62$ cases, many of them involves multiple records. If a case involves $N$ records, it is split into $\binom{N}{2}$ pair-wise cases.
Predicting Method

- For **Support Vector Classifier**, the output is the probability of the input record falling into group 1. Group 1 means positive, and group 0 means negative.

- For **Support Vector Regression**, the output is simply the regression result.

- For **Random Forest**, the output is also the probability of the input record falling into group 1.

Evaluation Method

We use both in-sample test and leave-one-out cross-validation.

- In **in-sample test**, all 96 cases are used to in the machine learning algorithm training session, and we use the trained model to predict the 96 cases one by one. We have 1 trained model here.

- In **leave-one-out cross-validation**, we use 95 cases to train the model, and use that model to predict the remaining 1 case. Do this for all 96 cases one by one. So we have 96 trained model in this evaluation method.

Comparison Results - In-sample Test

Since random forest does not have the same result each time, table 4.1 and table 4.2 show the results in two "in-sample test" experiments.

<table>
<thead>
<tr>
<th></th>
<th>SVR</th>
<th>SVC</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>thersholds</td>
<td>0.4</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>truepos</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>falsepos</td>
<td>4</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>falseneg</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.1: In-sample test result example 1
**Comparison Results - Leave-one-out Cross-validation**

Table 4.3 and table 4.4 show the results in two "Leave-one-out Cross-validation" experiments.

<table>
<thead>
<tr>
<th></th>
<th>SVR</th>
<th>SVC</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>thersholds</td>
<td>0.2</td>
<td>0.1</td>
<td>0.25</td>
</tr>
<tr>
<td>truepos</td>
<td>15</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>falsepos</td>
<td>11</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>falseneg</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.3: Leave-one-out cross-validation test result example 1

<table>
<thead>
<tr>
<th></th>
<th>SVR</th>
<th>SVC</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>thersholds</td>
<td>0.4</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>truepos</td>
<td>17</td>
<td>17</td>
<td>15</td>
</tr>
<tr>
<td>falsepos</td>
<td>4</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>falseneg</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.2: In-sample test result example 2

**Comparison Results**

In the tables 4.1, 4.2, 4.3 and 4.4, we can find that:
Support Vector Regression (SVR) performs slightly better than Support Vector Classifier (SVC). Both SVR and SVC's performances are deterministic, while Random Forest (RF) has some randomness (as its name has implied), generating different results if the program is run multiple times. This favors the Support Vector Regression method.

- The thresholds in two evaluation methods differs less than SVR or SVC. This favors the Random Forest method.

Thus we use the score produced by Support Vector Regression as the primary ranking score, and use the score generated by Random Forest as a reference score.

### 4.4 False Positive Rate Reduction

Figure 4-1 and figure 4-2 show that the introduction of Situation Comprehension Layer helps to reduce false positive rate a lot. SCL can learn from existing labeled false positive, and thus could identify similar cases and would not report them as positive cases.
4.5 Score Distributions

Another improvement brought by Situation Comprehension Layer is the wider spread and finer differentiation of the risk scores.

There are 3555 Red Flag Networks after the Integration Layer (NIL) processed the red flags raised by the Flag Layer (FAL). Table 4.5 and table 4.6 show the risk score distributions of Ranker Layer (RAL) and Situation Comprehension Layer (SCL).

<table>
<thead>
<tr>
<th>bin</th>
<th>0</th>
<th>0.01</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>counts</td>
<td>0</td>
<td>0</td>
<td>2532</td>
<td>368</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bin</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>counts</td>
<td>0</td>
<td>0</td>
<td>509</td>
<td>137</td>
<td>11</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 4.5: Risk score distribution of RAL for PRC FY2012 data

<table>
<thead>
<tr>
<th>bin</th>
<th>0</th>
<th>0.01</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>counts</td>
<td>2789</td>
<td>1</td>
<td>191</td>
<td>519</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bin</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>counts</td>
<td>7</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>11</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 4.6: Risk score distribution of SCL for PRC FY2012 data

Figure 3-4 and figure 3-5 shows the treemap view of the RFNets with risk scores generated by RAL and SCL respectively.

Compared with the scores generated by RAL, those generated by SCL tend to spread out better. This is non-surprising, because SCL’s input has more degree of freedom.

4.6 False Positive Examples

Usually, one transaction corresponds to one invoice, which should have a unique invoice number, and two documents with the same invoice number should be duplicate
payments. However, there are many reasons in reality that makes two documents with the same invoice number NOT duplicate payments.

4.6.1 Department-generated Invoice Numbers

Sometimes, the payee cannot provide a invoice number after receiving the payment, so the department generated one. This will result in lots of "duplicate" invoice numbers, whereas they are not duplicate payments. This may because:

The payee is a person that works as contract worker for the department.

Example: Two workers, Burke and Barry, worked for the Commission for the deaf and hard of hearing (MCD) as interpreters. They had the same hourly rate compensation, and worked for a same meeting, so they were paid the same amount. Since they could not give invoice numbers in return to the payment, the department used the payment date as the invoice number. Hence, the only difference between these two payments is the vendor profiles.

Luckily, we know that object "JJ2" is used for "Auxiliary Services" like this, we can suspect similar cases if we find a document’s "object" is also "JJ2". Please note that the vendor may also use date as part of their invoice numbers, wo we cannot disregard all invoice numbers that contains a date.

The payee gets batched (e.g. monthly) reimbursement from the department.

Example: Sterling Adult Day Care Inc. and Cheshire Adult Day Care Inc. received $278.64 in two separate documents. The payments are part of the state's monthly reimbursements to Adult Day Health centers for federal USDA Child Adult Care Food Program (CACFP) funds. The payments are generated based on a fixed rate per meal served, and those two Day Care Inc.s served the same amount of meals in that month, so they were paid the same amount.
The payee is a public entities (such as the government, library or hospital) that receives fund from the government.

Example: The City of Chelsea and the City of Lowell both received a same amount of $450,000 from the Executive Office of Health and Human Services (EHS). Since they are public entities instead of commercial vendors, they cannot provide invoice numbers as well.

The object class "PP", which represents "Grants and Subsidies", usually has this issue, so we may identify similar cases by checking if a document’s object class is "PP".

Another character of such false positive cases is that, there are usually many documents have the same invoice number, as the grant is usually awarded to multiple recipients.

4.6.2 Separated Invoices

An invoice can be intentionally separated into several documents. This may because:

The payment is split over several recipients

Example: Department of Developmental Services (DMR) bought 5 sheds from Home Depot for 5 different group homes residences, so DMR created 5 documents with the same invoice number.

The payment is split over different encumbrances

Example: Department of Public Health (DPH) made 2 $937.5 payments to a medical associate from 2 different encumbrances. It turns out that 1 was for administrative services, and the other was for clinical services.

In general, having different encumbrance numbers does not necessary exclude a pair of documents from true positives. There is another example where the Military Division (MIL) accidentally paid oil company A when they should have paid oil company B. The payment to company A is a duplicate payment, and the pair of
records is true positive that we want to identify. In this example, the two documents also have different encumbrances.

**The payment has more than maximum allowed number of lines**

Example: Mass Rehabilitation Commission (MRC)’s payments to Easter Seals Massachusetts are required to listed out by consumers, so there are a lot individual payments. Since MMARS (Massachusetts Management Accounting and Reporting System) only allows 99 lines per PRC document, MRC has to break out these invoices into multiple PRCs.
Chapter 5

Contributions of the SAFARI Platform

5.1 Increased Levels of Intelligence

The SAFARI platform can quantify risks for large volumes of heterogeneous data - labeled and unlabeled - thanks to its many components that form an intelligent integral.

The link analysis component reveals hidden relationships from different aspects. The Bayesian Belief Network component is dedicated to work with unlabeled data, and the machine learning techniques in the Situation Comprehension Layer is used to learn from examined incidents.

5.2 Reduced False Positive Rate

As briefly stated in section 4.4, the cases reported by SAFARI have a significantly lower false positive rates than current fraud detection methods, especially the duplicate payments detection methods.
5.3 Flexibility

SAFARI is an open platform that can be extended to take advantage of new techniques and add more tools in data and visual analytics.

5.3.1 Use Existing Framework Blocks

Users can build their own tools easily with existing framework blocks.

For example, if a user wants to find groups of different vendors on record that are likely to be the same vendor, or that are likely to be associated with the same business, he or she only needs to modify the detector and ranker configurations.

The existing framework blocks entitles users to find suspicious relationships by simply enter the corresponding configuration in User Interface.

5.3.2 Construct New Components

New components are also not hard to build, which facilitates the future upgrading of SAFARI platform. In this way, users can easily take advantage of new advances and add more capabilities in data and visual analytics.

Each layer of the platform has simple interface to upstream and downstream layer, as illustrated in section 2.2.1, which makes adding new components a easier task.

For example, if a user wants to construct vendor profiles as suggested in section 5.4.1, he can: 1) add a database to store vendor profile in DAL, and construct vendor profiles as he likes in data enrichment process; 2) add new detectors to FAL to raise red flags at individual document; 3) use existing NIL to integrate RFNets; 4) slightly modify the ranker at RAL to define the risk score of RFNets based on both the link risk scores and the node risk scores; 5) add features associated with vendor profile in SCL.

As a flexible software framework, SAFARI can keep up with business evolution and continue to be a valuable advisor in decision-making in both the short and the long term.
5.4 Future Work

5.4.1 Construct Vendor Profiles

In the scenario example in chapter 3, we emphasized on link analysis between pairs of document. We may also be interested in some anomalies related to a single document, such as out-of-order invoice serial number, or higher-than-peer charge. In this case, a "vendor profile" has to be constructed, so that we understand what is "normal invoice serial number", or what is a vendor’s "peer".

From RFNet’s perspective, we calculate both every link’s risk score and every node’s risk score with probabilistic analysis, and combine them to define a RFNet’s risk score.

5.4.2 Other Financial Deceptions

The SAFARI platform can easily be used in other types of financial deceptions, such as credit card and health insurance fraud.

In fact, we experimentally used SAFARI to examine medical reimbursement fraud, and it gave useful insights.

5.4.3 Risk Ranking for Decision Makers

SAFARI can adapt to a variety of environments where the assessment of risk is critical for decision making, such as the risk assessment of scale formation in oilfield facilities, computer cyber-security analysis, and the health, safety, or environmental incidents.
Bibliography


