Kieran Sears

MSc Commercial Games Development  
University of The West of England  
Faculty of Environment and Technology  
Computer Science and Creative Technologies Department

2018

Investigation into requirements of a financial simulation to assist the training of an artificial neural network

## Abstract

## Table of contents

Contents

[Abstract 1](#_Toc533874120)

[Table of contents 2](#_Toc533874121)

[Chapter one: Introduction 3](#_Toc533874122)

[Context 3](#_Toc533874123)

[Problem being addressed 3](#_Toc533874124)

[Dissertation aims 3](#_Toc533874125)

[Structure of this report 3](#_Toc533874126)

[Chapter two: Research 4](#_Toc533874127)

[Language 5](#_Toc533874128)

[Framework 5](#_Toc533874129)

[Data 7](#_Toc533874130)

[Assumptions / Drawbacks 8](#_Toc533874131)

[Chapter : System Architecture 10](#_Toc533874132)

[Chapter : Evaluation 10](#_Toc533874133)

[Chapter : Conclusion 10](#_Toc533874134)

[Summary of project 10](#_Toc533874135)

[Summary of results 10](#_Toc533874136)

[Discussion to the extent to which aims have been met 10](#_Toc533874137)

[Discussion of the extend to which the problem has been solved 10](#_Toc533874138)

[Future work 10](#_Toc533874139)

[Bibliography 10](#_Toc533874140)

## Chapter one: Introduction

### Context

Debt collection companies categorise customers to apply rule-based strategies to reconcile their customers arrears and ultimately get them out of debt in the least evasive way. Because categorising customers is a multi-dimensional space with many identifiers for a given customer it lends itself to ML techniques to calculate which category a given customer will fall into in turn with which strategy is best for category.

Before ML can be employed to such a sensitive domain, evidence advocating its viability and reliability is necessary to avoid putting irrevocable stress on vulnerable customers who reasonably cannot afford to repay their debt.

Simulations may be able to provide supportive evidence in a cost-effective way to the alternative of performing live trials. If ML is employed without supportive evidence the consequences of it malfunctioning will reflect on the businesses which use it, the customers who are incorrectly handled, the economy and confidence in ML as a tool for positive change.

The additional benefit of proving a simulation as a means of validating strategies is that the domain can be gamified to train employees of debt collection agencies against a wide variety of conditions as an educational training platform. This would increase the competence of staff and allow them to learn from actions which in the daily environment of their office would result in real consequence to a demographic more likely to be deemed vulnerable, making it more ethically conscientious.

### Problem being addressed

The problem being addressed for this project is discerning what requirements are necessary in developing a prototype system which could handle the demands of an enterprise scale financial simulation which in turn can be used to evaluate a machine learning algorithm, namely a neural network, so that meaningful information can be extracted from and extrapolated upon in order to be of use in decision making for a given scenario under a number of conditions.

In this way a more formal scheme could be devised and built upon to produce something of merit to the financial debt collections industry to guide decision making as the use of machine learning becomes more prevalent.

### Dissertation aims

Assess the suitability of the Actor model paradigm

Assess the suitability of the Spark framework used in this context

Assess the use of LSTM RNNs for the classification of customers

### Structure of this report

## Chapter two: Research

### Language

Things to consider: Ease of use, interoperability / cross platform compatibility, code complexity, concise notation, testability, libraries available

#### Scala

* Already familiar with the language
* Works on JVM so provides cross platform
* Similar syntax to Java with lambdas so quite easy to understand
* Can use the same libraries as Java which are already well established
* Provides code complexity optimization
* Offers concise notation
* Functional programming style means functions lend themselves to black box testing
* Provides pattern matching
* Good documentation
* Allows OOP design like object/class, inheritance, interface and polymorphism

Scala matches on most of the things to consider and also ticks one or two other boxes to boot.

### Framework

#### Akka

Akka provides clustered event sourcing and Command Query Responsibility Segregation (CQRS) architecture to separate concerns and be able to attach multiple views to the system, for example having a display to view the results of the learning algorithms for analysis, or having a game make calls to the system to calculate what the next actions a player can take are. It ensures a web service has no one single point of failure. And maintains an event log which provides a full history of events that have passed through a system. Snapshots or periodic state saves of the system can be replayed from this event log and used for system recovery if a complete failure was to occur.

Akka can scale out to multiple JVMs which means horizontal scaling is easy to achieve and is managed entirely by akka configuration which dynamically handles loads. If this project prototype proves successful it could later expand as part of the roadmap to collect data from many sources simultaneously, which for use of ML and big data would be advantageous. For example, the ML models used in the server could be getting data from hundreds of employees at a time if the client was implemented as a game or training simulator ran for a medium to large loans company.

However, being distributed will require considering:

Interface needs to be simple to use and has a well-defined contract to follow.

Have concurrent computation to deal with demand / computation complexity of ML training.

compromise between consistency, availability or partition tolerance (CAP Theorem).

What kind of configurations are available as part of the framework. E.g. recovery strategies.

The Actor Model [1: Carl Hewitt]:

Actors are fundamental units of computation and embody: processing, storage and communications. Each Actor can create more actors, send messages and designate how it handles it’s next message. Actors communicate through messages which they process one at a time, but can queue into a mailbox (FIFO message queue) messages if it is already in the middle of processing a different message.

Actors are in a hierarchy structure, whereby parent Actors manage children by a process called supervision. This ensures that upon a fatal internal actor failure, that actors parent can detect the failure start a new Actor in its place.

Actors have an address which is how they are referenced when we want to send one a message. These are integrally maintained through encryption, so that addresses cannot be generated and introduced from outside the scope of the Actor hierarchy tree. An address is not an identity though, as we could have “one address for many Actors” by having one actor with many addresses who acts as an intermediary and orchestrates forwarding to other actors who actually do the processing.

Messages operate by a “best efforts” policy, which entails messages being persisted locally, so resending can be initiated upon failure of a message not sending successfully, but if the node the sending actor is on goes down the local copy is lost and cannot be recovered, hence messages can may be completely lost.

There are no channels between actors, messages are shared more like packets over a network matching on the Actors address. This means intermediaries can be avoided because of costly overheads in operations such as two-phase commits, whereby two actors may need the same message. Messages are sent at most once unless specified to do otherwise by a custom time-out strategy.

Actors work with Futures which are a place-holder object for a value which may not yet exist. Composing concurrent tasks in this way tends to result in faster, asynchronous, non-blocking parallel code because of the use of callbacks in place of blocking operations. [5]

Actor model provides concurrent computation and more convenience in programming. It outsources the overhead of implementing multi-thread safe code through use of actors in place of threads which implement message queues (FIFO) in the form of mailboxes. State is kept local/isolated within each actor until propagated explicitly via messages meaning intercommunication between cores doesn’t require shared memory or atomic data structures. Fault tolerance is addressed through actor supervision according to a configurable recovery strategy thereby providing a self-healing system.

“Easy” parallel processing can be achieved by splitting tasks that meet Bernstein conditions and having the constituent sub tasks sent via multiple messages to actors and have the supervisor system handle horizontal scaling dynamically without the need for complex code.

Cons to remember:

deadlocks

overflowing mailboxes

In the actor model consistency is the sacrificed element to the CAP theorem. This was the chosen sacrifice because systems are inherently inconsistent due to the physics of the speed of signals between components anyway, so it is best to focus efforts on the areas that can be more rigorously controlled. [Inconsistency robustness 2011, 2014]

[1] Carl Hewitt; Peter Bishop & Richard Steiger (1973). "A Universal Modular Actor Formalism for Artificial Intelligence". IJCAI.

[2] Bernstein, A. J. (1 October 1966). "Analysis of Programs for Parallel Processing". *IEEE Transactions on Electronic Computers*. EC-15 (5): 757–763. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.1109/PGEC.1966.264565](https://doi.org/10.1109%2FPGEC.1966.264565).

[3] Report to Brewer’s CAP Theorem CS341 Distributed Information Systems Salomé Simon University of Basel, HS2012

[4] Martin Fowler

[5] <https://docs.scala-lang.org/overviews/core/futures.html>

#### Apache Spark

Apache spark is a unified analytics engine. It is a powerful open source engine that provides real-time stream processing, interactive processing, graph processing, in-memory processing as well as batch processing with very fast speed, ease of use and standard interface.

### Data

#### Input

Customers are one of the primary inputs into the simulation. They contain fields which define them as an entity and from these fields can be mapped the expected behaviour that customer will take if there is no intervention. These behaviours will affect the metrics of the simulation in the following way.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customer Type | Time Taken | Cost | Return | Happiness |
| Self-cure | Short | Very Low | Full | Medium |
| Arrangement successful | Long | Low | Full | Medium |
| Arrangement unsuccessful settlement | Long | Medium | Part | High |
| Arrangement unsuccessful litigation | Long | Very High | Full | Low |
| Settlement | Short | Medium | Part | High |
| Litigation | Long | High | Full | Low |

Customer Types in how they will affect the state of the system

The fields that decide these behaviours will include:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Customer Type | Sex | Age | Disposition | Arrears | Happiness | Income |
| Self-cure | M: 50 F:50 |  |  |  |  |  |
| Arrangement successful | M: 50 F:50 |  |  |  |  |  |
| Arrangement unsuccessful settlement | M: 50 F:50 |  |  |  |  |  |
| Arrangement unsuccessful litigation | M: 50 F:50 |  |  |  |  |  |
| Settlement | M: 50 F:50 |  |  |  |  |  |
| Litigation | M: 50 F:50 |  |  |  |  |  |

Normalisation of customer characteristics

#### Output

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Action | Self-cure | Arrangement successful | Arrangement unsuccessful settlement | Arrangement unsuccessful litigation | Settlement | Litigation |
| Send letter |  |  |  |  |  |  |
| Phone call |  |  |  |  |  |  |
| Write off |  |  |  |  |  |  |
| Litigate |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Agent Actions

|  |  |
| --- | --- |
| Action name | Effects |
| Add Customers | State statistics, state customer list, |
|  |  |
|  |  |
|  |  |

System Actions

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Customer Actions

### Machine Learning Algorithm

#### Quick overview on the evolution of Neural Networks

##### Perceptron

##### Multi-layered perceptron

##### Deep-learning algorithms

##### Recurrent neural networks

Vanishing gradient problem = parameters become biased towards short-term dependencies.

e.g. modelling language, trying to guess a penultimate word in a sentence “In France, I had a great time and I learnt some of the \_ language.” The algorithm would be negatively bias towards the word France, which disadvantages the algorithm in connecting the missing word to the answer “French”.

Solutions around this: Choice in activation function – RelU prevents shrinking gradients from the chaining backpropagation (prevents f’ function from shrinking the gradients). Initialisation of weights – if weights are initialised to an identity matrix, biases are initialised to zero, which prevents weights from shrinking the gradients.

Gated Cells:

1. Forget irrelevant parts of the previous state
2. Selectively update cell state values
3. Output certain parts of the cell state

How this helps:

1. The forget gate allows information to pass through unchanged, i.e. important inputs are kept and not multiplied by tiny weights each time
2. Sj depends on Sj-1 through addition, so back propagating doesn’t create a huge chain rule product

##### Long short term memory networks

### Assumptions / Drawbacks

* There is a pattern which maps customer attributes onto the type of behaviour they choose.
* That Agent Actions on customers influence their attributes and thereby behaviours.
* Generating data will map onto real world data with similar results.
* The attributes chosen for customers are reflective of real world data collected on customers.
* Customer credit scores and account history are not considered.
* How changes in config during runtime handle tasks already queued from previous config?

## Chapter : System Architecture

#### SBT

SBT build tool is used to manage library dependencies. Dependencies are stated in the build.sbt file at the root of the project folder structure. This helps in resolving dependency issues.

#### Akka

Akka is imported using SBT and is used for both implementation of the actor model as well as the HTTP interface to allow the project to run as a server to be called by external programs. The idea behind using the HTTP service from Akka other than for the compatibility it shares with the actor system, is to make the view of the model, view, controller architecture employed separate from the logic that governs the simulation and machine learning algorithm.

Akka also provides libraries for database integration allowing persistence of data for the project. Cassandra

## Chapter : Evaluation

## Chapter : Conclusion

### Summary of project

### Summary of results

### Discussion to the extent to which aims have been met

### Discussion of the extend to which the problem has been solved

### Future work

## Bibliography

Glossary