CHAPTER 4

THE DESIGN AND DEVELOPMENT OF THE PROTOTYPE

4.1 Introduction

The designed conceptual model of RiCCA as described in Chapter 3 is need to be further tested with a larger size of dataset to verify its feasibility. A prototype that is developing computationally in this study is following the guideline from the immunology simulation framework proposed by Figueredo, Siebers, Aickelin, & Foan (2012). This has been discussed in Section 3.2.

Previous studies have developed prototypes that using AIS algorithm for computational purposes. Some samples of prototype developed in AIS are including libtissue (Jamie Twycross & Aickelin, 2010) and LISYS (Hofmeyr & Forrest, 2000) or known as Lightweight Intrusion detection SYStem. Both of these prototypes are applied and focused for intrusion detection study.

The processes involved in RiCCA designation and development is depicted in Figure 4.1.

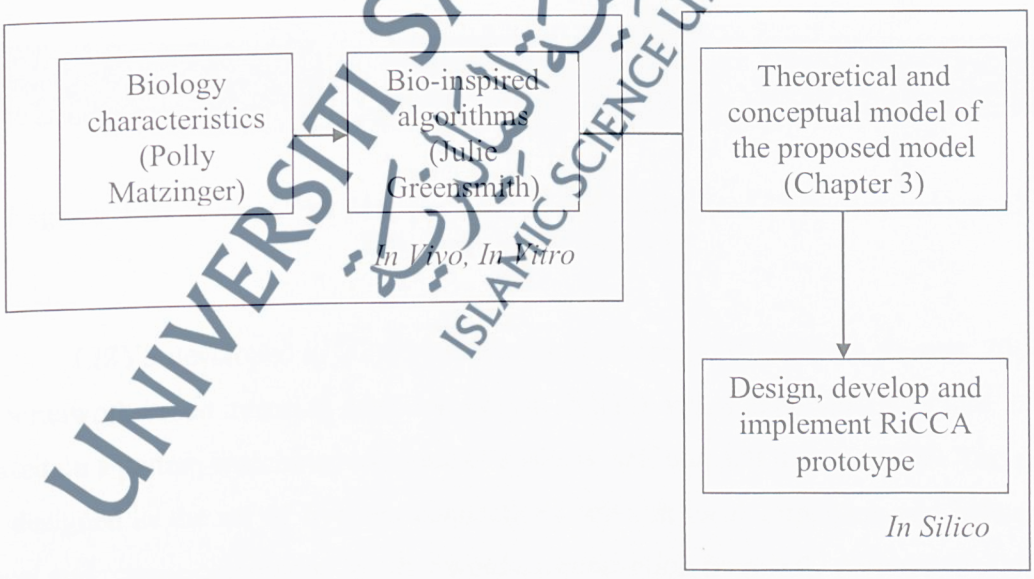

Figure 4.1: The Connection Of Biological Studies With RiCCA

Scientific research usually releases a prototype which intended to represent a working system of an idea rather than a theoretical one. This research is objectively to produce a working or a form of prototype, which is modelling a risk assessment product that is implemented and act as an assistant in decision making. This product is imitating the human body defense system. Illustrated in Figure 4.2, a concept mapping is represented between the biological perspectives of Human Immune System (*in vivo*), computational theory, Artificial Immune System (AIS), and the real implementation of AIS in computer security field, text spam risk assessment (*in silico*).

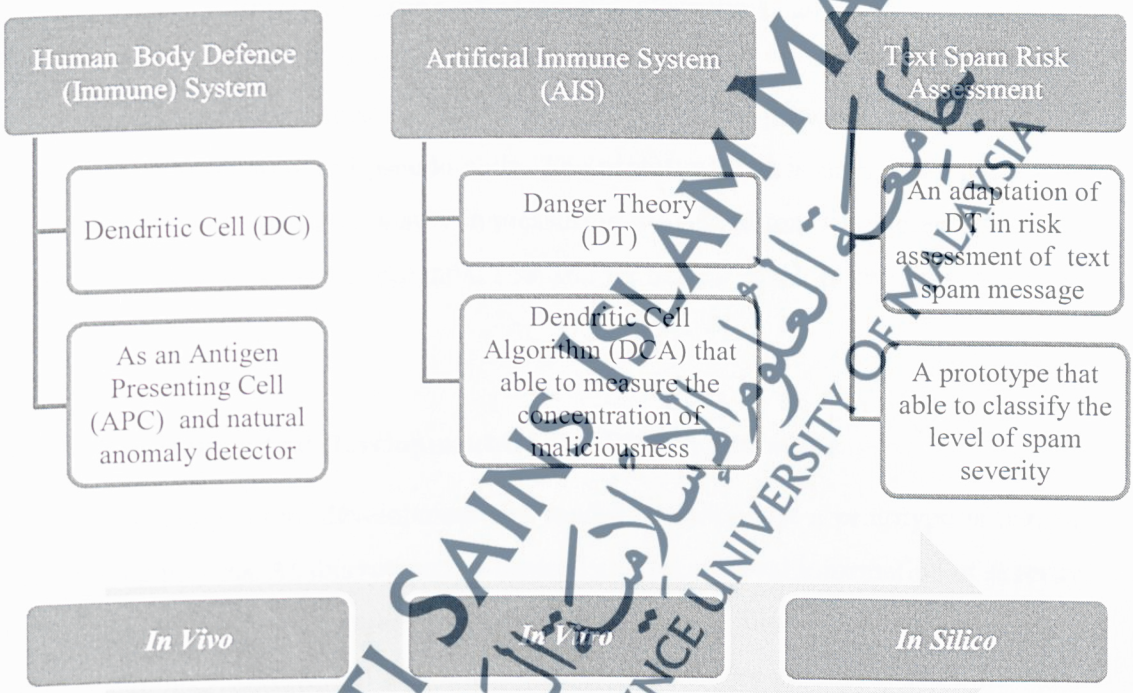

Figure 4.2: A Concept Mapping Between Biological (Theoretical) And The Developed Model

LISYS developed in 2000 by Hofmeyr and Forrest (Hofmeyr & Forrest, 2000) is a network-based intrusion detection system (NIDS) with a distributed structure. It is based on a pattern-matching technique of analysis, self non-self theory of AIS. The *self* is designed as the set of frequent connections between local computers and between local and remote computers. In other words, a connection frequently detected in a long period belongs to the *self*. The *non-self* is represented by rare connections. It is designed for anomaly detection that analyses the header of TCP/IP packets. LISYS is the most

important AIS for intrusion detection; in fact, many researchers (Gabrielli & Rigodanzo, n.d.; Kim et al., 2007) based their work on that system.

In 2010, Jamie Twycross & Aickelin has implemented an AIS algorithm library in C programming language that is based on properties of innate immunology. This is known as a libtissue, a prototype software system for constructing the second generation of AIS and applying them to real-world problems (J. Twycross & Aickelin, 2006). It is implemented as a library; algorithm can be compiled and run on other researchers' machines with no modification. Libtissue is used within the Danger Project (Aickelin et al., 2003) for testing of ideas and algorithms, as shown in works of Greensmith (2007) and J Twycross (2007).

This chapter includes the whole description of the prototype in terms of its algorithms (flowcharts) and pseudo-code. The prototype that is namely as RiCCA is developed based on its functionality in measuring the risk of text spam messages. This assessment is suggesting the potential risk and its impact level as an outcome of the RiCCA.

4.2 The Design And Development Of The RiCCA Prototype

The designation, development and model evaluation of a prototype is part of data mining process. As this research is basically about retrieval information of severity intensity from text spam message, a prototype is a must medium to execute this mining task. A summary overview of the flow sequence of the entire process is as follows and the detailed explanation about this process in practical data mining can be found in Monte F. Hancock (2012).

Step 1: Problem Definition

Step 2: Data Evaluation

Step 3: Feature Extraction and Enhancement

Step 4a: Prototyping Plan

Step 4b: Prototyping / Model Development

Step 5: Model Evaluation

Step 6: Implementation

An algorithm is normally described in a semiformal notation such as pseudo-code and flowcharts. Flowcharts are used mainly for the high-level description of the algorithms and pseudo-code for describing the details. Pseudo-code is a notation that uses a few simple rules and describes the algorithm that defines a problem solution. It can be used to describe relatively large and complex algorithms. It is relatively easy to convert the pseudo-code description of an algorithm to a computer implementation in a high-level programming language (Garrido, 2012).

In this research experiments, all the processes are elaborated via flowchart diagram; the pictorial representation of the whole logic and that will be further explained step by step. Then, an advance description of the algorithm is intricate via pseudo-code to illustrate the entire process. Eventually, the whole process will be developed as a set of the prototype, implemented using high-level programming language. The developed prototype of the model then will be evaluated in terms of its functional and performance.

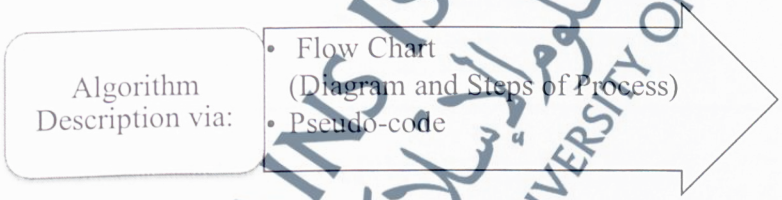

Figure 4.3: Processes Of Design And Development Of A Prototype

The developed RiCCA prototype is depicted in Figure 4.4 and articulated in Table 4.1.

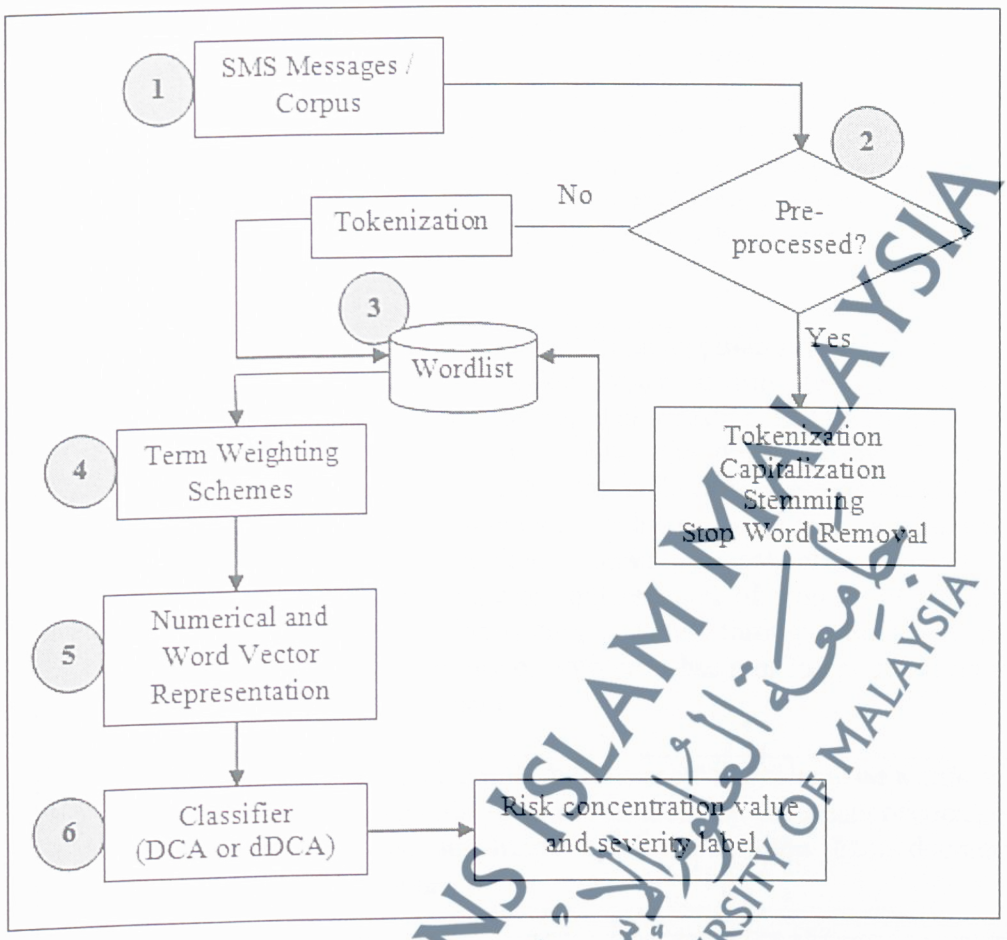

Figure 4.4: An Overview Of RiCCA Model

Table 4.1: Risk Assessment In Text Spam Messages

Phase	Process	Description
1	Preparation	Data, which consists of SMS messages (ham and spam) are collected and prepared for an initial database of the prototype. A new spam message can be processed once this initial database has been developed. Messages must be in text format such as .txt and .doc
2	Pre-processing	The corpus (for initial population) or text message have an option either to run through text pre-treatment or not. Tokenization is a must process if decided not to have the pre-processing in place. Otherwise, the full cycle of text pre-processing will be executed, which include tokenization, capitalization, stemming or also known as lemmatization and removal of stop word from the data. The testing that runs through either with pre-processing or without it has a different effect on the final result.
3	Wordlist	This wordlist database contains tokenized words with the information of its total and document occurrences and its frequency in spam and ham document category.
4	Term Weighting Schemes	Term Frequency (TF), Information Gain Ratio (IG Ratio) and Chi-square (CHI^2) are the available term weighting schemes and act as feature selection methods. These schemes will calculate the importance of every word in the corpus that indicates its relevance to the spam (risk) category. The value derived statistically from this method is further implemented as an input signal in the classifier. This information is stored as an internal database library. All schemas calculate tokenized word as in between 0 to 1 which suggested that the closer the value to 1, the closer it is to malicious level.
5	Numerical and Word Vector Representation	Calculated value for every term in the previous phase then will be mapped to the risk scale. Every derived value will represent the term's severity degree, for example, *PAMPs, Danger* and *Safe*. The range for risk scale is user-defined.

Table 4.1, continued

Phase	Process	Description
6	Classification	This phase consists of three (3) sub-process which are: • Signal processing – identified input signals in a message will be correlated and processed depends on the chosen classifier, either DCA or dDCA; • Context assessment – the content of the message is assessed for its risk concentration level in numerical value; and • Risk flagging – classified risk level, for example, *high, medium, low* will be marked to SMS message accordingly. The weightage scale used to differentiate input signals and output signals (risk levels) are referring to Table 4.3 and 4.4 for DCA and Table 4.5 for dDCA.

The following chapter will elaborate on prototype processes which cover the general process, the function of classifier DCA and dDCA, pre-processing of the corpus/message and term weighting schemes. These processes of designation and development of a research prototype would include flowchart, and pseudo-code as depicted in Figure 4.3. The source code for the prototype programming written in JAVA is attached in Annexure D.

To show the interrelationship between the proposed models of RiCCA as depicted in Figure 4.4 with the generic algorithm of DCA as reviewed in Chapter 3 (Algorithm 3.1), the following Table 4.2 portrayed all the related integration that is involved in the designation. The different colours of connections indicate a different phase in RiCCA model.

Table 4.2: The Relationship Portrayed For Integration Of DCA Algorithm In RiCCA Model

Process Flow (Figure 4.4)		The connection	Algorithm 3.1		
Phase	Process		Description	Process	Generic DCA
1	Preparation		collect dataset corpus	Create initial population	Line 4-13
			text pre-processing		
2	Pre-Processing		assign input signals value for antigen		
3	Wordlist		store antigen value		
4	Term Weighting Schemes		receive new spam message	Calculate the severity level	Line 20
			text pre-processing		
5	Numerical and Word Vector Representation		map the identified antigen with the stored input signals value (refer library database)		
			calculate the severity level using classifier (DCA or dDCA)		
6	Classification		map the calculated output signals with developed risk scale and identify the level of risk	Prioritize risk	Line 21
			according to identified risk level, response against spam could be delete, escalate to authority body, re-calculate the risk level (for the case of false positive), or do nothing	Action to response	Immune response activated

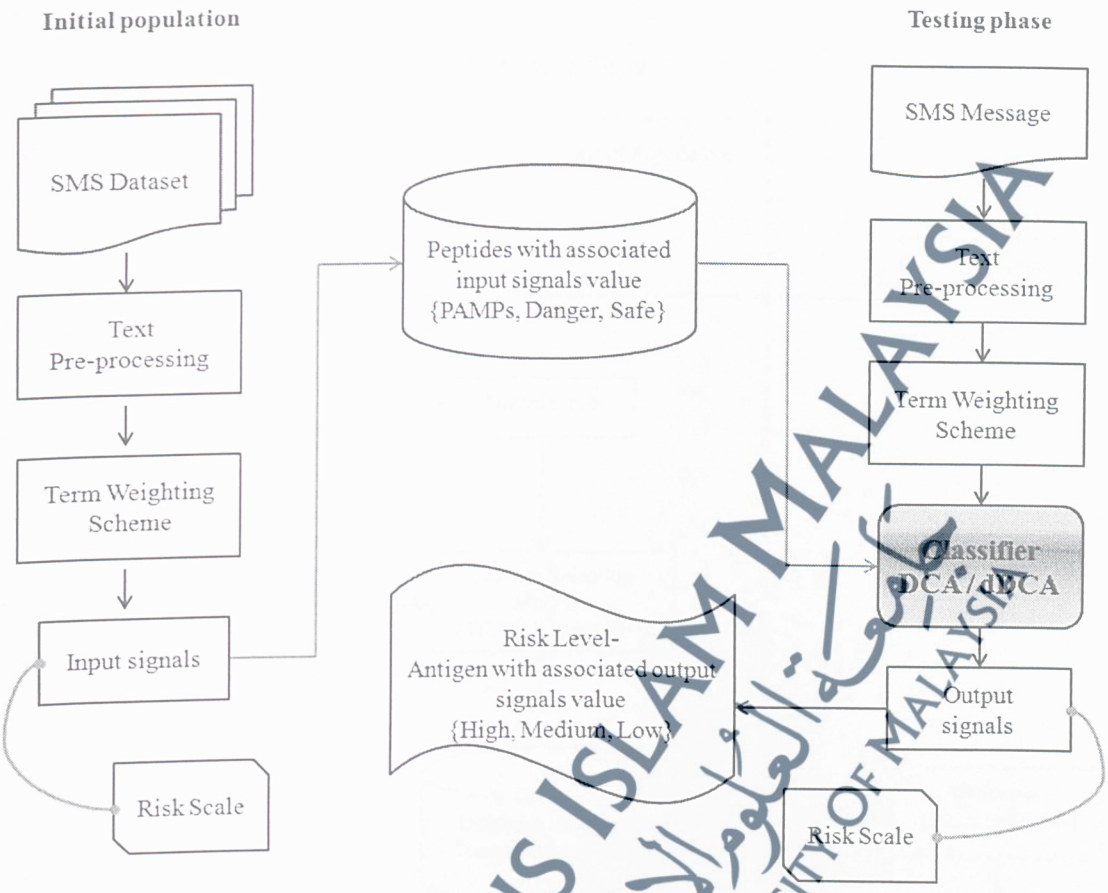

Figure 4.5: An Overview Diagram For The Entire Process Of RiCCA

The entire process for RiCCA model is illustrated in Figure 4.5. This diagram fundamentally exhibited two (2) eminent phases; initial population for the antigen and signals value preparation and the testing phase for new incoming SMS message. The integration of algorithm's classifier with the text mining processes is also presented. The text mining elements are including pre-processing and term weighting scheme. Risk scale is applied to define the level of input and output signals associated with the value pre-determined in risk scale. The same risk scale is also utilized to conclude the level of measured risk.

All of the above-mentioned processes are elaborated in detail in the following sections.

4.2.1 General Process

4.2.1.1 Diagram And Process Flow

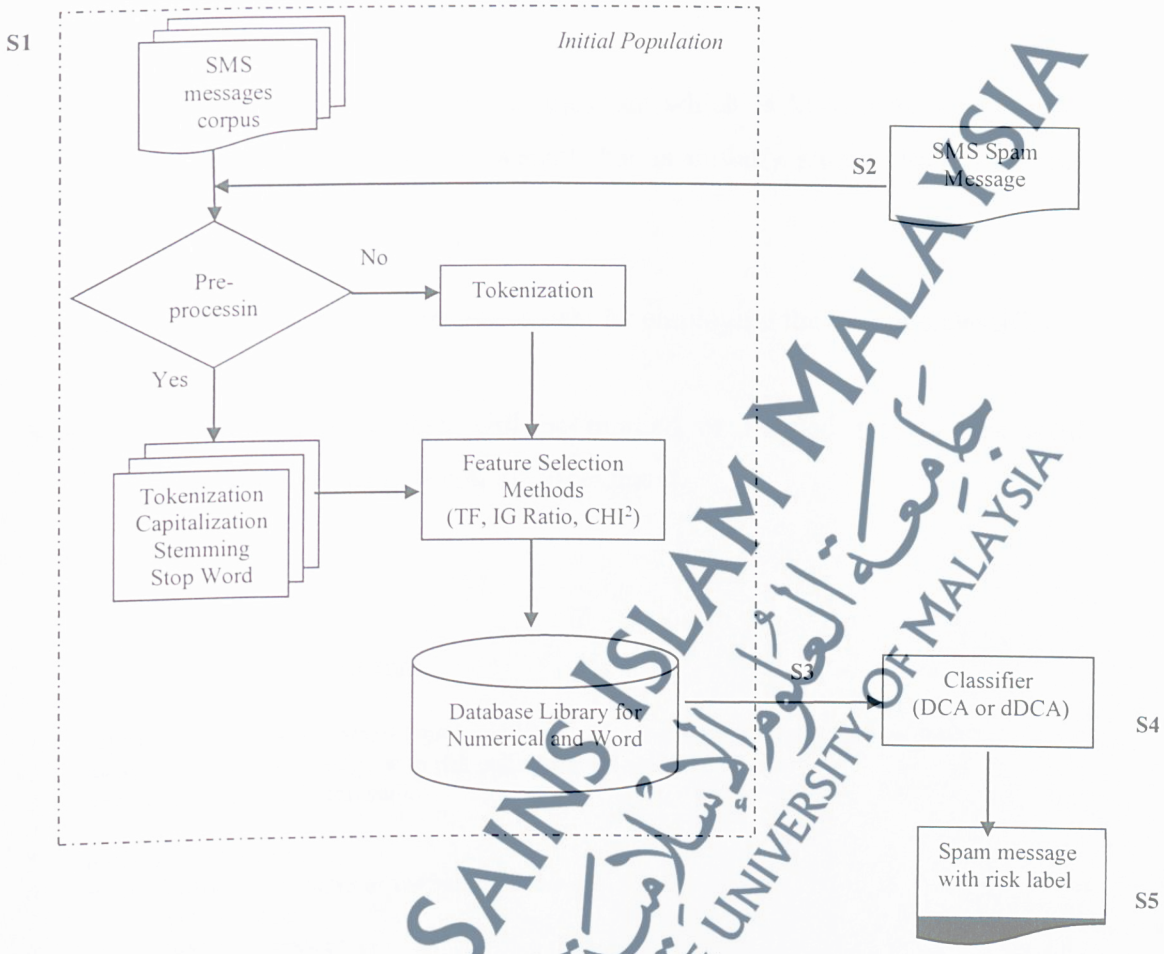

Figure 4.6: Diagram And Flow For General Process

Step 1 : Initial Population - Initialize antigen and signal pool to the library by processing the corpus that consists of both ham and spam messages. All messages at least need to be tokenized. The weight assigned for every term is determined and dependent on how the weighting schemes function. These chunked of antigen or peptides in numerical value are stored in the database library. Basically, this initial population is referring to process phase number 1 to 5 as depicted in Figure 4.4.

Step 2 : Initialize a DC (immature state) when a new spam message (antigen) flow into the model for processing. There is an option to run the message with or without pre-processing phase.

Step 3 : Map each term (tokenized message which is known as a peptide in immunology) with the weight that is initially stored (Step 1) in the database library.

Step 4 : Calculate the context assessment by employing the immune classifier.

Step 5 : Spam message then will be marked or flagged for its risk level accordingly with its context assessment.

4.2.1.2 Algorithm

```
1   input      : message dataset (spam and ham) and pre-categorized signals (weights)
2   output     : spam message with risk concentration value
3   parameter  : anomaly threshold
4
5   initialize assessment;
6   while dataset of messages or corpus available do
7           get messages
8           tokenize message
9           calculate weights for input signal generation
10          store assigned weights for every token
11  endwhile
12
13  for every new spam message do
14          tokenize message
15          choose weighting scheme
16          map every term with its numerical values
17          measure context assessment using classifier
18          print spam message with the calculated risk concentration value and its label accordingly
19  end
```

Algorithm 4.1: General Process of RiCCA

4.2.2 Pre-processing Of The Corpus/Message

4.2.2.1 Diagram And Process Flow

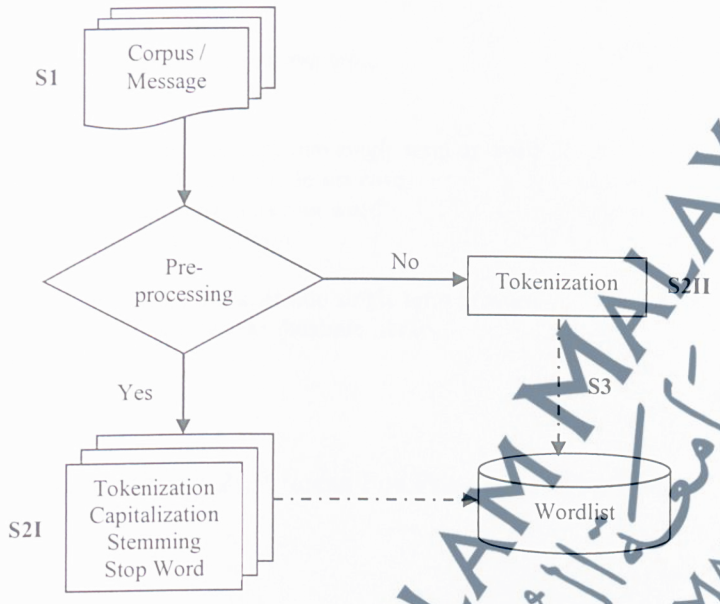

Figure 4.7: Diagram And Flow For Pre-processing Phase

Step 1 : To begin the process, a set of data or corpus need to be collected and prepared.

Step 2 : I. Execute the full pre-processing stages which include tokenization, capitalization, stemming and removing stop word.

II. If choose not to execute the pre-processing, data must be at least tokenized, the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens.

Step 3 : The pre-processed tokens then further fed into weighting schemes prior to being stored as input signals.

4.2.2.2 Algorithm

```
1   input       : message dataset (spam and ham)
2   output      : tokenized message
3
4   initialize text processing;
5   while dataset of messages or corpus available
6           if with pre-processing
7                   then
8                           split every message into single term or word
9                           change all capital to lower case
10                          convert word to its root word
11                          remove all stop word
12                  else
13                          split every message into single term or word
14                  store all tokens into the database library
15          endif
16  endwhile
```

Algorithm 4.2: Process For Pre-processing Phase

4.2.3 Term Weighting Schemes As Feature Selection Methods

4.2.3.1 Diagram And Process Flow

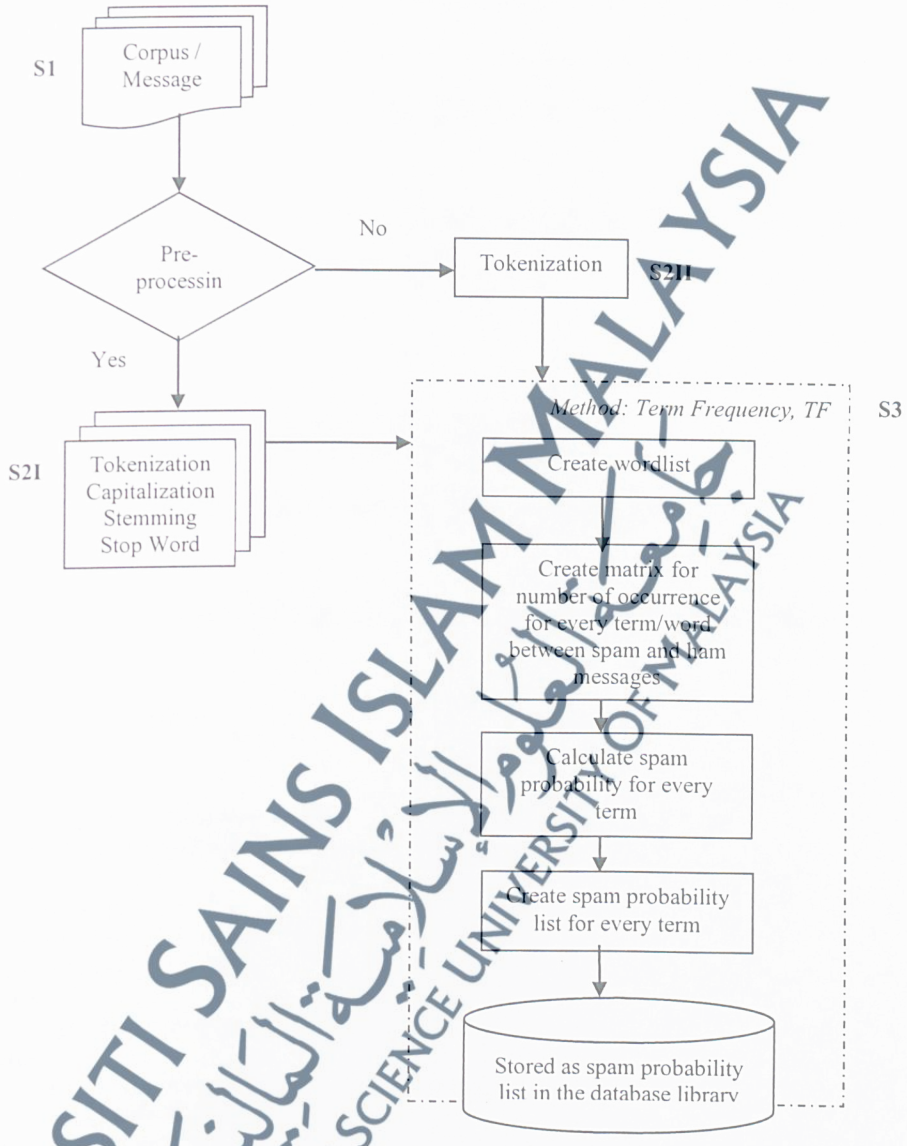

Figure 4.8: Diagram And Flow For Term Frequency, TF Process

The figure only demonstrated the calculation of Term Frequency (TF). TF also referred as the strength of a term in spam category. The IG Ratio and CHI^2 values are derived from the RapidMiner.

Step 1 : To begin the process, a set of data or corpus need to be collected and prepared.

Step 2 : I. Execute the full pre-processing stages which include tokenization, capitalization, stemming and removing stop word.

II. If choose not to execute the pre-processing, data must be at least tokenized, the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens.

Step 3 : The tokenized word then further fed into weighting schemes (TF) for calculation its spam probability, S_p value.

$$S_p(\textbf{term}) = \frac{frequency\ of\ \textbf{term}\ occurence\ in\ spam}{total\ frequency\ of\ \textbf{term}\ occurence\ in\ all\ messages} \quad (4.1)$$

*all messages refer to total number of spam and ham messages

Step 4 : The calculated weight for every word then stored in the database library.

4.2.3.2 Algorithm

```
1   input       : message dataset (spam and ham)
2   output      : list of spam probability value for every term
3
4   initialize text processing;
5   while dataset of messages or corpus available
6       if without pre-processing then
7           then
8               split every message into single term or word
9           else
10              split every message into single term or word
11              change all capital to lower case
12              convert word to its root word
13              remove all stop word
14      endif
15      for Term Frequency (TF) do
16          create wordlist
17          create a matrix for a number of occurrence for every term/word between spam and
            ham messages
18          calculate spam probability for every term
19          create spam probability list for every term
20      endfor
21      store spam probability value for all term into the database library
22  endwhile
```

Algorithm 4.3: Process For Term Frequency, TF

4.2.4 Dendritic Cell Algorithm (DCA) Process

4.2.4.1 Diagram And Process Flow

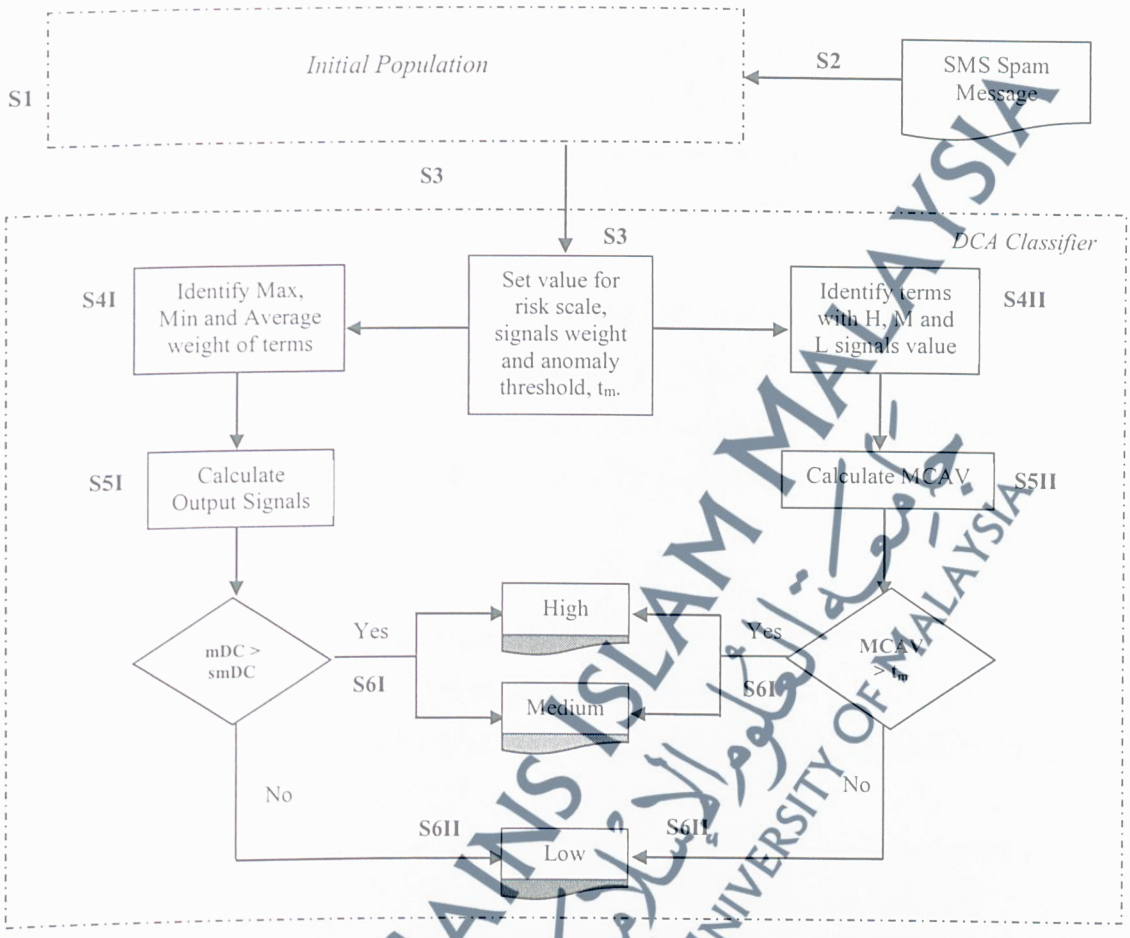

Figure 4.9: Diagram And Flow For DCA Classifier Process

Step 1 : Initialize antigen and signal pool by collecting SMS messages/corpus that consists of both ham and spam messages. The messages entry is recorded as an anomaly threshold value, t_m (Equation 4.2). This t_m value is further utilized in Step 5II.

$$Anomaly\ threshold, tm = \frac{No.of\ spam\ messages}{Total\ messages\ (ham+spam)} \qquad (4.2)$$

Then, execute text pre-treatment with or without pre-processing. All messages must at least need to be tokenized. The weight assigned for every term is determined and dependent on how the chosen weighting schemes

function and the antigen with its calculated weight value are stored in the database library.

Step 2 : Initialize a DC (immature state) when a new spam message (antigen) flow into the model for processing. There is an option to run the message with or without pre-processed stage. When this new antigen instances available, they will be further processed and added to the database library.

Step 3 : Set value range for risk scale (Table 4.3) and weight (Table 4.4) to transform an input signal to output signal. Set value for t_m as recorded in Step 1. Then, map the term (tokenized message which is known as a peptide in immunology), that initially stored (Step 1) in the database library with the defined risk scale.

Step 4 :

I. Identify maximum, minimum and median (average) signal value for every level (PAMPs, Danger, Safe) of an antigen; and

II. Identify terms with high, medium and low signals value (refer Table 4.3).

Table 4.3: Proposed Scale Of Risk Level Value, In Between 0 To 1

Signal Value	Range of Value (Input Signals)	Level of Risk, MCAV (Output Signals)
In between 0 to 1	PAMPs	High
	Danger	Medium
	Safe	Low

Step 5 :

I. Calculate the output signal, *O [CSM, smDC, mDC]* for that particular antigen by correlating the input signals (refer Equation 4.3 to 4.5 and Table 4.4). The output signal is calculated for three (3) cycles to represent the maximum, minimum, and average value of the signals; and

Output signals:

$$O[CSM]=(W_{p1}*\text{PAMPs}) + (W_{d1}*\text{Danger}) + (W_{s1}*\text{Safe}) \quad (4.3)$$

$$O[smDC]=(W_{p2}*\text{PAMPs}) + (W_{d2}*\text{Danger}) + (W_{s2}*\text{Safe}) \quad (4.4)$$

$$O[mDC]=(W_{p3}*\text{PAMPs}) + (W_{d3}*\text{Danger}) - (W_{s3}*\text{Safe} \quad (4.5)$$

Table 4.4: Proposed Weight Matrix For Signal Correlation Value

Input signals	PAMPs	Danger	Safe
CSM	W_{p1}	W_{d1}	W_{s1}
smDC	W_{p2}	W_{d2}	W_{s2}
mDC	W_{p3}	W_{d3}	$-W_{s3}$

The calculation of output signals must be done in three (3) cycles to demonstrate the maximum, minimum and average signal value. For example, the calculation for the cycle of maximum value, first cycle as follows:

$O_{max}\ [CSM] = (W_{p1} * PAMPs_{max}) + (W_{d1} * Danger_{max}) + (W_{s1} * Safe_{max})$

$O_{max}[\ smDC] = (W_{p2} * PAMPs_{max}) + (W_{d2} * Danger_{max}) + (W_{s2} * Safe_{max})$

$O_{max}\ [\ mDC] = (W_{p3} * PAMPs_{max}) + (W_{d3} * Danger_{max}) - (W_{s3} * Safe_{max})$

The calculation is continuing for average value as for the second cycle and minimum value for the third or last cycle. This output signal value from 3 cycles (maximum, minimum and median value) then compared between *O[mDC]* and *O[smDC]*. Refer Step 6 to determine the severity or normality of the message.

II. Calculate the MCAV value (refer Equation 4.6). Mature counts refer to the number of words with high and medium signals value. Then, compare MCAV with anomaly threshold, t_m.

$$MCAV\ (antigen\_type) = \frac{mature\_count}{antigen\_count} \qquad (4.6)$$

Step 6 : Merge all the findings (in Step 4 and 5) to get a final assessment of antigen context. Match and compare to each other for the measured values.

I. $O[mDC] > O[smDC]$ and MCAV > t_m - this is to indicate that spam message is malicious and MCAV result should return a value as High or Medium; or

II. $O[mDC] < O[smDC]$ and MCAV < t_m - this is to indicate that spam message is benign and MCAV result should return a value as Low.

These conditions are based on discussion by authors Greensmith & Aickelin (2008) and Greensmith (2007), which has been articulated in Table 2.6.

4.2.4.2 Algorithm

```
1   input       : message dataset (spam and ham) and pre-categorized signals (weights)
2   output      : spam message with risk concentration value
3   parameter   : anomaly threshold
4
5   initialize assessment;
6   while dataset of messages available do
7           get messages
8           tokenize message
9           calculate weights for signal generation
10          store assigned weights for tokenized words
11  endwhile
12  for every new spam message do
13          tokenize message
14          choose weighting scheme
15          set value for risk scale with three (3) different levels (Table 4.3) and weight (Table 4.4)
            to transform input signal to output signal
16          identify for the maximum, minimum and average value of weights
17              calculate OUTPUT SIGNALS, O[CSM, smDC, mDC] for three (3) cycles;
                maximum, minimum and average signals value (refer Equation 4.3-4.5).
18              if O[mDC] > O[smDC] then,
19                      spam message is assigned as 1 to indicate the malicious message; and
20                      MCAV result should calculate as High or Medium (refer Table 4.3)
21              else
22                      spam message is assigned as 0 to indicate the benign message; and
23                      MCAV result should calculate as Low (refer Table 4.3)
24          identify words with high, medium and low signals value (refer Table 4.3)
25              calculate MCAV (refer Equation 4.6)
26          identify anomaly threshold, t_m (refer Equation 4.2)
27              if MCAV > anomaly threshold, t_m then,
28                      spam message is anomalous
29              else
30                      spam message is normal
31              endif
32              print spam message with the MCAV concentration value and its associated risk level
33  end
```

Algorithm 4.4: Process For DCA Classifier

Original algorithm for DCA can be found in Greensmith, Aickelin, & Cayzer (2010), and Greensmith (2007) or Algorithm 2.1 in Section 2.5.2 of this thesis.

4.2.5 Deterministic Dendritic Cell Algorithm (dDCA) Process

4.2.5.1 Diagram And Process Flow

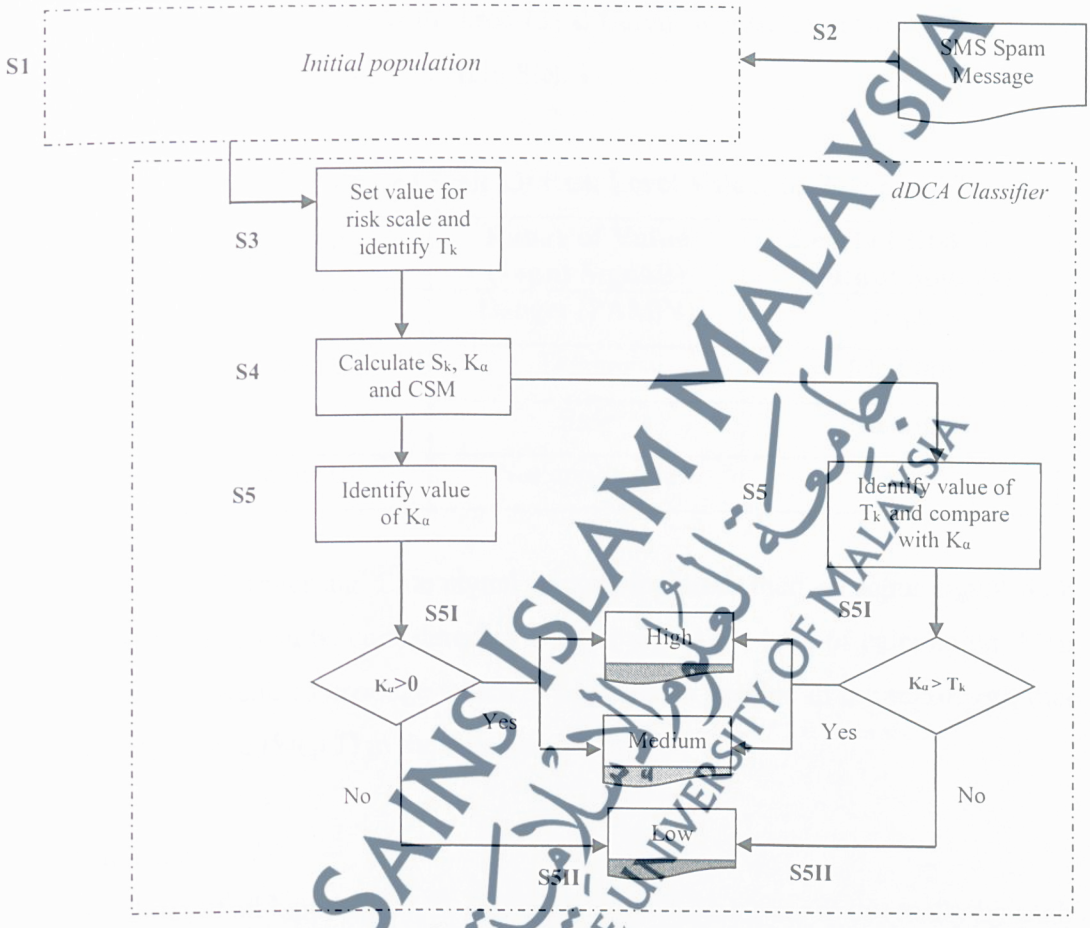

Figure 4.10: Diagram And Flow For dDCA Classifier Process

Step 1 : Initialize antigen and signal pool to the library by processing the corpus that consists of both ham and spam messages. The messages entry is recorded as an anomaly threshold value, T_k (refer Equation 4.2, equal as t_m). This T_k value is further utilized in Step 5. All messages at least need to be tokenized. The weight assigned for every term is determined and dependent on how the weighting schemes function.

Step 2 : Initialize a DC (immature state) when a new spam message (antigen) flow into the model for processing. There is an option to run the message with or without

pre-processed phase. When there are new antigen instances, they will be further processed and added to the database.

Step 3 : Set value for risk scale with three (3) different levels, in between 0 to 1 and identify value for T_k as recorded in Step 1.

Table 4.5: Proposed Scale Of Risk Level Value, In Between 0 To 1

Signal Value	**Range of Value (Input Signals)**	**Level of Risk, K_α (Output Signals)**
In between 0 to 1	Danger (PAMPs)	High
	Danger	Medium
	Safe	Medium
Below 0	Not applicable	Low

Since only Danger and Safe signal is contemplated, then an input signal with PAMPs value can be considered as Danger signal for ease of calculation. Map the term (tokenized message which is known as a peptide in immunology), that initially stored (Step 1) in the database library.

Step 4 : Calculate:

- Sum of all input signal, S_k (refer Equation 4.7), k (stored internally by each DC);

  $k = D-2S$

  $S_k = \sum D - 2 \sum S$ (4.7)

  D is danger signal, S is safe signal.

- K_α, magnitudes of k value (refer Equation 4.8);

  $K_\alpha = S_k / \alpha_m$ (4.8)

  K_α generates real value anomaly scores and may assist in the polarization of normal or anomalous process, α_m number of antigen presented.

- Anomaly threshold, T_k (refer Equation 4.2)

The similar threshold value can be derived from the MCAV using the ratio of total danger signals to total all signals presented in the used dataset (Greensmith & Aickelin, 2008).

Step 5 : Identify the value of K_α (positive value indicate anomalous and negative value indicate normal) and compare with the value of T_k. For this output signal, value Safe in risk scale is considered as Medium risk since; the Low risk level is negative value, below 0 (refer Table 4.5). This requires both of the following conditions are fulfilled.

I. $K_\alpha > 0$ and $K_\alpha > T_k$ - spam message is tagged as the malicious message; or
II. $K_\alpha < 0$ and $K_\alpha < T_k$ - spam message is tagged as the benign message.

These conditions are based on discussion by authors Greensmith & Aickelin (2008) and Greensmith (2007), which has been articulated in Table 2.6.

4.2.5.2 Algorithm

```
1   input      : message dataset (spam and ham) and pre-categorized signals (weights)
2   output     : spam message with risk concentration value
3   parameter  : anomaly threshold
4
5   initialize assessment;
6   while dataset of messages available do
7           get messages
8           tokenize message
9           calculate weights for signal generation
10          store assigned weights for counted words
11  endwhile
12  for every new spam message do
13          tokenize message
14          choose weighting scheme
15          set value for risk scale in between 0 to 1
16          calculate :
17              • sum of all input signal, Sk (refer Equation 4.7), k (stored internally by each DC)
18              • Ka, magnitudes of k value (refer Equation 4.8),
19              • Threshold, Tk (refer Equation 4.2)
20
21              if Ka > 0 and Ka > Tk then,
22                      spam message is tagged as the malicious message; and
23              else
24                      spam message is tagged as the benign message; and
25              endif
26              print spam message with it's tagged label (normal or anomalous)
27  endfor
```

Algorithm 4.5: Process For dDCA Classifier

Original algorithm for dDCA can refer Greensmith & Aickelin (2008) or Algorithm 2.2 in Section 2.7.

4.3 Summary

This chapter basically illustrates the details of process flow via multiple diagrams and algorithms. The steps are explained in details including the mathematical equations involved. This will help the development process in writing the prototype in a programming language. The prototype's reliability is required to be validated with a series of experiments and hence to confirm the flow is well functioning. The design and development of the prototype are basically following the immunology simulation framework proposed by Figueredo, Siebers, Aickelin, & Foan (2012). This is further tested with a larger size of the dataset and the prototype implementation is referring to the simulation part which is explained in Chapter 5.