UNIVERSITY OF NAIROBI

COLLEGE OF BIOLOGICAL AND PHYSICAL SCIENCES

SCHOOL OF COMPUTING AND INFORMATICS

AUTOMATED CUSTOMER SUPPORT WITH CONVERSATIONAL AGENTS EMPLOYING TEXT MINING

A Case of Online University Application

BY

MUNYIVA MBITHI NGEA

P58/76753/2012

SUPERVISOR: DR AGNES WAUSI

December, 2014

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

DECLARATION

The project, as presented in this document, is my original work and has not been presented for any other university award.

Signature :________________________ Date:______________________________

Ngea Munyiva Mbithi

P58/76753/2012

The Project has been submitted in partial fulfillment of the Requirements for the Degree of Master of Science in Computer Science at the University of Nairobi with my approval as the University Supervisor.

Signature :________________________ Date:______________________________

Dr Agnes Wausi

School of Computing and Informatics

ACKNOWLEDGEMENT

I wish to thank God for granting me with the ability to complete this project and the following people for their contributions;

My supervisor **Dr. Agnes Wausi** for her guidance and encouragement in undertaking the project from conception to completion.

My partner **Geoffrey M. Mimano** for his support, encouragement and resources that facilitated conclusion of this project.

My parents **John Mbithi** and **Elizabeth Mwikali** for instilling the values of hard work and determination in me, and **Charles** and **Grace Mimano** for their support and facilitation in the course of this project.

The rest of my family for their encouragement, support, facilitation and prayers during the course of this project.

The University Online application staff for their facilitation and provision of data that was used in this study.

Lastly my friends and colleagues for their facilitation, ideas and interest in the project.

ABSTRACT

The use of online information systems that can be and are accessed on a 24 hour basis has continued to grow both locally and globally, these systems require available customer support at all times. Customer support is services that assist customers to make cost effective and correct use of products or services; it is divided into voice-based and non-voice based. Requests received by support personnel are routine in nature and their numbers surpass the staff employed to handle them hence organizations need to device mechanisms of automating customer support to ensure available staff deal with requests that have been escalated due to their uniqueness as opposed the routine ones; thereby enhancing productivity and reducing staff related costs and improving the response times.

This study demonstrates automation of customer support, through design and development of an automated customer support system that uses the online university application as a case study. The system employs conversational agents (CA) which are computer systems that are intended to converse with human beings. The demonstrated CA use classification algorithms specifically Naive Bayes (NB) algorithm and Latent Semantic Indexing (LSI) to categorize the emails received.

The developed application can process and respond to 500 emails in 3 minutes, while support personnel can handle the same in approximately 5000 minutes, hence demonstrating its applicability in possibly reducing staff costs and improving response times. A comparative analysis of responses generated was performed on a subset of the sample support emails to determine the accuracy of the application as compared to the customer support personnel. NB algorithm was found to have an error rate of 88 percent while LSI was found to have an accuracy of 74%. Further evaluation conducted on the algorithms resulted in LSI accuracy 0.89 and F-Score 0.94 while NB accuracy 0.94 and F-Score 0.96. The study recommends that any future studies to study the improvement of accuracy of the NB classifier and system enhancement to be able to handle multi- part emails.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	- Artificial intelligence
CA	– Conversational Agents
CBR	- Case-based reasoning
FAQs	- Frequently Asked Questions
GO-CA	- Goal-Oriented Conversational Agents
IE	- Information Extraction
JAB	– Joint Admission Board
LSA	- Latent Semantic Analysis
LSI	- Latent Semantic Indexing
KNN	- K-Nearest Neighbor
KUCCPS	– Kenya Universities and Colleges Central Placement Service
NB	- Naïve Bayes
NLP	- Natural Language Processing
POS	- Part-of-Speech
STI	- Science, Technology and Innovations
SVD	- Singular Value Decomposition
SVM	- Support vector machines

1 INTRODUCTION

1.1 Background

The rapid growth of the internet, intranets, extranets and other interconnected global networks in the 1990s dramatically changed the capabilities of information systems in business. Globally internet-based and web-enabled enterprise and global electronic business and commerce systems have become commonplace in the operations and management of business enterprises. The internet and related technologies and applications have changed the way businesses operate and people work, and how information systems support business processes, decision-making and competitive advantage. Today many businesses are using internet technologies to web-enable business processes and to create innovative e-business applications(Chow, 2009). In Kenya private companies adopt the latest in online information systems technology in order to maintain a competitive edge while government agencies and institutions driven by Kenya Vision 2030 that proposes intensified application of Science, Technology and Innovations(STI) to raise productivity and efficiency (GoK, 2007) have slowly began to move operations that were previously handled manually online to web based information systems. Examples of government agencies and institutions that have implemented online information systems in the recent past include; Kenya Universities and Colleges Central Placement Service (KUCCPS) the former Joint Admissions Board (JAB) in application for admission into public universities and colleges, Kenya Revenue Authority (KRA) in filing of tax returns, Kenya Civil Service in recruitment of staff, and the Higher Education Loans Board (HELB) in student loan application. Other institutions that have been using online information systems include Educational institutions in student management, job applications and student application. The government online information systems are accessed by thousands of users, consequently increasing demands on the customer support provided. These online services require the availability of customer support on a 24 hour basis as customers access the applications at different times at their convenience.

Customer support is a range of services that assist customers in making cost effective and correct use of a product, it includes assistance in planning, installation, training, troubleshooting, maintenance, upgrading and disposal of a product (BusinessDictionary.com, 2014). Customer support is divided into two categories: voice-based customer support which involves customers calling and asking for assistance and non-voice customer support services which are differentiated into chat support, email support, (Open Access BPO, 2013) and more recently social media based support whereby customers post their inquiries or problems on social media.

In Kenya both forms of customer support are employed, with voice-based support being more popular as evidenced by the number of organizations which are setting up and which already have existing contact-centers these include banks, mobile network operators and digital content providers. The popularity of voice-based support has been supported by the drastic growth in the mobile industry that resulted in an increased number of mobile subscribers approximately 31 million in December 2013 (CCK, 2014) up from 15,000 in 1999 (CCK, 2008) and the lowering of calling costs in the industry. With the landing of the three undersea fiber-optic cables at the Kenyan Coast and laying of terrestrial fiber-optic network across the country, the cost of accessing internet has drastically reduced while speed and reliability of internet connectivity improved (Mokaya, 2012), consequently the number of internet users has increased as illustrated by Table 1 which compares the total number of internet users of Kenya in 2007 and in 2013.

Table 1: Internet Users in Kenya

Indicator	**Sep 2007**	**Dec 2008**	**Dec 2009**	**Sep 2010**	**Sep 2011**	***Dec 2012***	**Dec 2013**
Internet users (cumulative)	2,865,646	3,359,552	3,648,406	8,689,304	14,300,679	16,236,583	21,273,738

Source (CCK, 2014)

As more people access the internet and familiarize themselves with various web applications they also relay their support requests on non-voice based platforms these include; email, chat and social media. The growing move to non-voice based customer support platforms has forced organizations to hire more personnel to respond to requests that are received on the various channels. In Kenya some of the organizations that offer customer supports on email as well as social media include; Safaricom a mobile network operator, Kenya Power a utility provider, Kenya Airways an airliner and University of Nairobi.

Effective customer support and service has become a strategic imperative, regardless of the organization’s field of specialization, what is increasingly making a competitive difference is the customer support and service that is built into and around the product or service, rather than just the quality of the product or service (Sawy and Bowles, 1997). Hence organizations have to ensure that the customer support offered is provided in a timely manner and it is helpful to customers. With many of the customer requests received on a daily basis being routine in a nature, an organization has to find ways of automating the handling of such requests to ensure available staff deal with requests that have been escalated as opposed the routine ones; thereby enhancing productivity.

Lester, Branting & Mott (2004) studied conversational agents (CA) which are computer systems that are intended to converse with human beings. These conversational agents integrate computational linguistics techniques with the communication channel of the Web to interpret and respond to statements made by users in ordinary natural language. The study highlighted the use of the agents in the areas of customer service, help desk, website navigation, guided selling and technical support (Lester et al., 2004). The conversational agents were used to automate customer support by simulating dialogue between customer and customer support personnel.

Text mining is the process of extracting patterns from natural language rather than from structured database facts, employing algorithms for converting unstructured text into structured data and conveying the insightful information (Segall et al., 2009). The patterns extracted must be valid and potentially useful. Text mining was used to categorize requests received from customers into the various predefined categories that determined the action or response to be given.

1.2 Problem statement

The growth of social media platforms, use of email, and chat for business processes has resulted in a move to non-voice based customer support. To remain competitive organizations ensure that all platforms are manned by support staff who read, interpret and handle customer requests appropriately, this has resulted in large number of employees employed for the sole role of providing support. With the increase in employees organizations are faced with challenges specifically training costs, providing competitive remuneration and staff turnover. The organizations in the face of these challenges should implement methodologies to automate customer support in order to decrease the staff requirements.

Most customer requests are routine in nature and can be handled automatically without need for human intervention, example of these requests include; queries on how to use an application, queries whose responses already exist in frequently asked questions (FAQs) website page and queries whose responses are determined by database data. In addition as the number of users accessing online applications dramatically surpasses staff employed to provide support, and as applications become more complex then, the need for efficient handling of requests is emphasized in order to ensure that customers can effectively navigate and use applications with ease. As we know humans as prone to err; as such some simple customer requests are sometimes misunderstood by support personnel resulting in customer frustration.

1.3 Purpose Statement

The purpose of the study was to design, implement and study and automated customer support system and demonstrate its effectiveness using the online university application as a case study.

1.4 Objectives

The following were the objectives of this study:

1. To identify and analyze techniques used in automating customer support and select suitable technique for creating a customized automatic customer support system.
2. To develop a customized automatic customer support system using the selected technique.
3. Conduct a comparative analysis of the performance of the system developed with human customer support.

1.5 Research Questions

1. Are there automatic customer support systems in the market? What are the methodologies used in their development?

2. What are the various issues that are encountered in customer support applications?
3. Do existing methodologies have gaps that can be addressed?
4. How can an integrated customized application methodology be developed to address identified gaps?
5. Can the methodology developed be applied in a real world performance problem scenario?
6. Does an automated customer system perform comparably with human in terms of time, accuracy?

1.6 Justification

Automated customer support system offers organizations running online applications accessed by thousands of users a mechanism to automate the handling of routine queries. In addition it extends possible support offered by 1) proving additional information to customers and 2) includes not only the issues raised but also those that have a likely hood of occurrence. In addition the system also dramatically reduces the response time and increases the possible volumes of requests that can be handled within a day, week or month.

1.7 Scope and Limitation

The study was restricted to the response of requests received via email and for testing purposes those placed locally in a mail directory. Due to the complexity of case online application and the multitude of possible issues and requests that can be received the study was restricted to a small facet of the application. The accuracy and efficiency of the prototype developed was also capabilities of natural language processing (NLP) technologies; specifically the categorization of issues which was heavily dependent on the algorithm used, availability of training data and “cleanliness” of the training data available. Some issues raised require human intervention hence there is an escalation component in the prototype developed.

2 LITERATURE REVIEW

This chapter focuses on previous work related to this study, and provides a theoretical background to the research work.

2.1 Customer Support

Customer support is a range of services that assist customers in making cost effective and correct use of a product, it includes assistance in planning, installation, training, troubleshooting, maintenance, upgrading and disposal of a product (BusinessDictionary.com, 2014). Customer support is divided into two categories: voice-based customer support which involves customers calling and asking for assistance and non-voice customer support services which are differentiated into chat support, email support, (Open Access BPO, 2013) and more recently social media based support whereby customers post their inquiries or problems on social media.

2.2 Automating Customer Support

Automation of customer support entails building a database of known issues and their resolutions with delivery mechanisms ("Customer support," 2014). Customer support automation has been studied and implemented through Artificial Intelligence (AI), a section of computer science that is concerned with making computers act rationally or intelligently. In AI the automation studied and implemented with; Case-based reasoning (CBR) which is a problem solving paradigm whereby new problems are solved by remembering previous similar situations and reusing the information and knowledge of that situation (Aamodt and Plaza, 1994), Conversational agents also referred to as dialog systems are computer systems intended to converse with humans with a coherent structure ("Dialog system," 2014).

Wang et al(2010) studied the use of CBR in intelligent help desk agents the study addressed the two challenges that pertain to CBR namely; case retrieval measures whereby most systems use traditional keyword-matching-based ranking schemes for case retrieval and have difficulty in capturing the semantic meanings of cases and result representation whereby most case-based systems return a list of past cases ranked by their relevance to a new request, and users have to go through the list and examine the cases one by one to identify their desired cases. To address these challenges the study developed iHelp, an intelligent online Helpdesk system, that automatically finds problem–solution patterns from the past customer–representative interactions. When a new customer request arrives, iHelp searches and ranks the past cases based on their semantic relevance to the request, groups the relevant cases into different clusters using a mixture language model and symmetric matrix factorization, and summarizes each case cluster to generate recommended solutions (Wang et al., 2011).

Chang et al(1996) studied SmartUSA a CBR helpdesk system developed for Union Camp Corporation that solved customer problems by filtering the problem description through an alias table to generate a brief description and then matching the brief description with the cases in the database. SmartUSA proved to be an effective and user friendly system that successfully handled different descriptions of the same problem and allowed for the case base to be built in free-format (plain) text. The system significantly reduced the workload and the response time in the customer services department. The study concluded that though impressive success had been achieved by SmartUSA, more sophisticated approaches for case representation, storage and indexing are needed for CBR based systems(Chang et al., 1996) .

While CBR has been successfully applied in customer care systems the main issues in its use are; The CBR system requires large storage space for all cases and it also requires large processing time to find similar cases. Also cases may need to be created by hand and needs case-base, case selection algorithm, and possibly case-adaptation algorithm. In short if one requires the best solution or the best optimum solution then CBR is not the best option. Most user support systems based on CBR are used to guide customer care personnel on possible diagnosis or solutions to problems as they display the most related cases; however this study proposes to develop a system that will automatically diagnose and resolve issues.

Lester, Branting & Mott (2004) studied conversational agents (CA) which are computer systems that are intended to converse with human beings, these conversational agents integrate computational linguistics techniques with the communication channel of the Web to interpret and respond to statements made by users in ordinary natural language. The study highlights the application of the agents in the areas of customer service, help desk, website navigation, guided selling and technical support. The study further elaborates on the requirements the conversational agents must satisfy; First, providing sufficient language processing capabilities that they can engage in productive conversations with users that is being able to understand users' questions and statements, employing effective dialog management techniques, and accurately responding at each conversational turn. Second, operating effectively in the enterprise that is must exhibit scalability and reliability and clean integration into existing business processes and enterprise infrastructure.

The Conversational Agents are viewed as alternative mechanism to mitigate the shortfalls in CBR systems, specifically the space and time requirements.

2.3 Conversational Agents (CA)

Accurate and efficient natural language processing (NLP) is essential for an effective conversational agent. To respond appropriately to a user's utterance, a conversational agent must (1) interpret the utterance through the interpreter, (2) determine the actions that should be taken in response to the utterance carried out by the dialog manager, and (3) perform the actions by response generator, which may include replying with text, presenting Web pages or other information, and performing system actions such as writing information to a database. Figure 1 illustrates the data flow in a conversational agent as described above while Figure 2 shows the natural language processing components of a conversational agent (Lester et al., 2004).

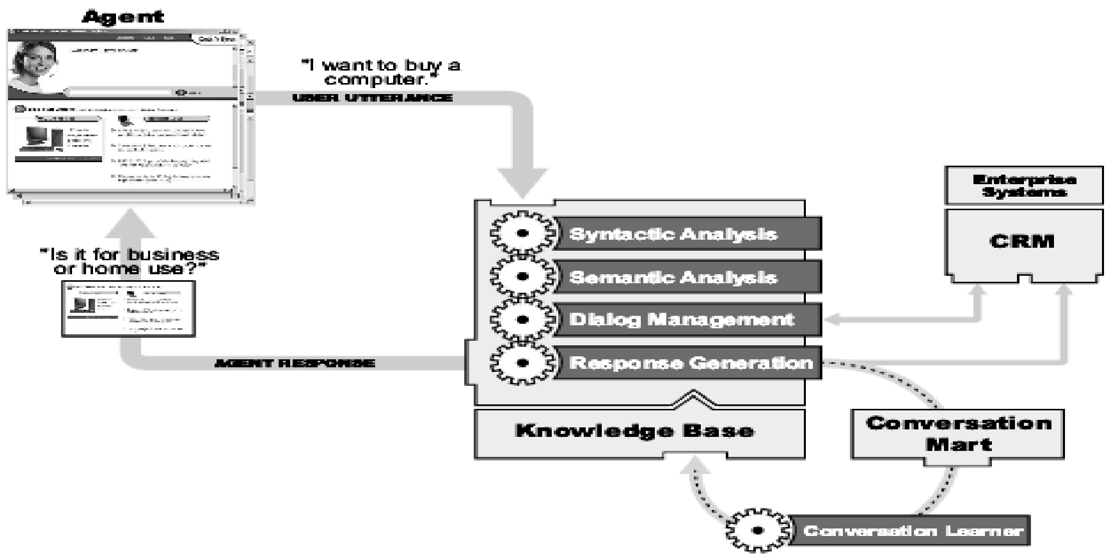


Figure 1: Data Flow in a Conversational Agent

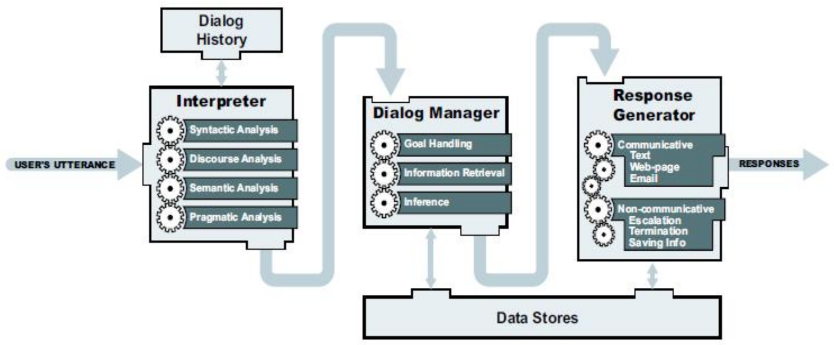
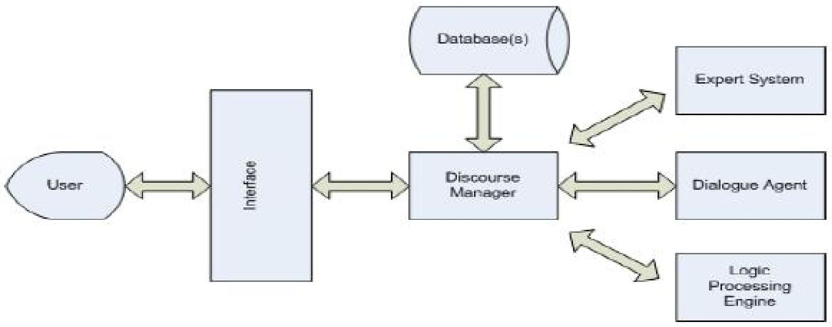


Figure 2: The primary natural language components of a conversational agent

Conversational agents offer a solution to the cost versus effectiveness tradeoff for customer support. By engaging in automated dialog to assist customers with problems conversational agents address inquiries at a much lower cost than human-assisted support. The conversational agents can only operated in circumscribed domain but can offer a cost-effective solution in applications where the requirements are bounded (Lester et al., 2004).

Crocket et al. (2011) studied goal-oriented conversational agents designed to converse with humans through use of natural language dialogue to achieve specific tasks. A Goal-Oriented CA (GO-CA) is a type of conversational agent which has a deep strategic purpose that enables it to direct a conversation to achieve a goal. The goal oriented conversational agents utilized pattern matching algorithms to capture the values of specific attributes through their dialogue interaction with a user. They achieved pattern matching through use of scripts that contain sets of rules about the domain and a knowledge base to guide the conversation towards a specific goal. These systems are stated to be ideal for providing clear and consistent advice 24 hours a day in; advising on organizational policies and procedures, guiding customers on products, and tutoring students to understand learning objectives. Figure 3 shows the architecture of a goal-oriented conversational agent (Crockett et al., 2011).



1

Figure 3: GO-CA Architecture

Feng et al., 2003 described WebTalk, a general framework for automatically creating spoken and text-based customer care dialog applications based entirely on organization's website. Webtalk leveraged on the vast information available on websites and enabled tight synchronization of dialog systems with website updates. Figure 4 illustrates the major components of WebTalk; Website analyzer constructs dialog oriented task knowledge from given websites taking websites as input and outputting task data, The speech recognizer derives a language model from website and structured task knowledge, Language understander is a rule-based component that converts natural language sentences into semantic representation, Dialog manager is the core of a dialog system and it involves more handcrafted work such as predicting possible dialog states, designing associated actions and responses for each state, lastly language generation is responsible for refining text output of the system to be dialog-style natural prompts. The paper demonstrated the automatic creation of dialog systems to be used in customer care (Feng et al., 2003) and its results can be leveraged when creating customer care systems that pertain only to website data.

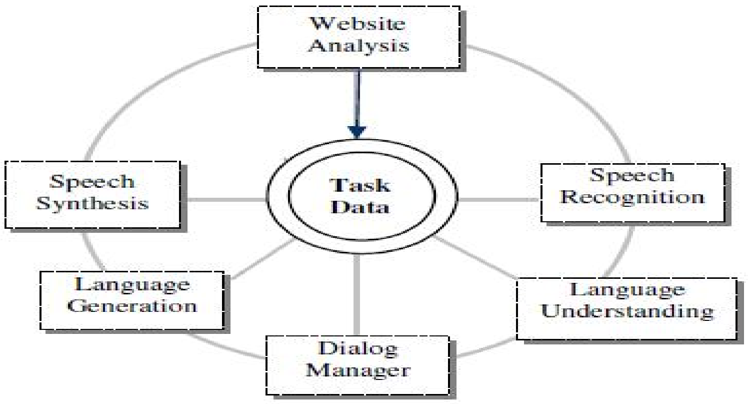


Figure 4: A Schematic Diagram of Major Technology Components in WebTalk

In order to satisfy the natural language processing requirement the study proposes using text mining categorization algorithms.

2.4 Text Mining

Text mining is finding useful data by automatically extracting information from textual sources examples of sources include; web documents, emails, database, and text messages. It is the process of extracting patterns from natural language rather than from structured database facts, employing algorithms for converting unstructured text into structured data and conveying the insightful information (Segall et al., 2009). The patterns extracted must be valid and potentially useful. The text mining process consists of the following stages illustrated by Figure 5;

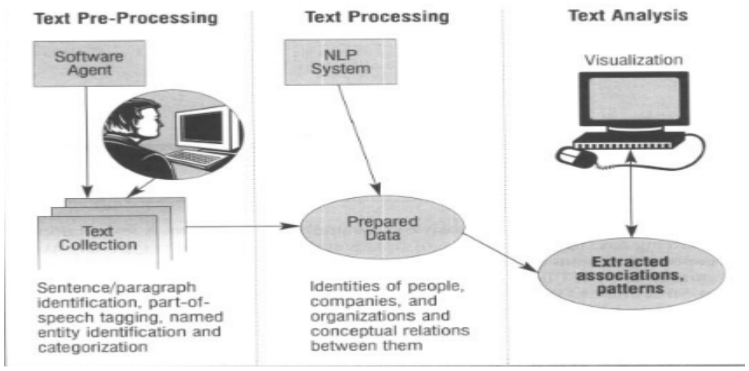

Text Pre-Processing
Text Processing
Text Analysis
Software Agent
NLP System
Visualization
Text Collection
Prepared Data
Extracted associations, patterns
Sentence/paragraph identification, part-of-speech tagging, named entity identification and categorization
Identities of people, companies, and organizations and conceptual relations between them

Figure 5: Text Mining Process

Source(Liddy, 2000)

2.4.1 Text Preparation

Text preparation or pre-processing is the selection, cleansing and pre-processing of text. In this stage selection of sources for text mining would occur, usually under the guidance of a human expert, and early text pre-processing, such as sentence identification and part-of-speech tagging, would take place (Liddy, 2000). The goal of pre-processing is to represent the text in such a way that its storage in the system and retrieval from the system is very efficient (Lama, 2013).

Text pre-processing includes the following stages:

2.4.1.1 Tokenization

Tokenization is the process of chopping up a given stream of text or character sequence into words, phrases, symbols, or other meaningful elements called tokens which are grouped together as a semantic unit and used as input for further processing such as parsing or text mining. Usually, tokenization occurs in a word level but the definition of the "word" varies accordingly to the context. So, the series of experimentation based on following basic consideration is carried for more accurate output(Lama, 2013):

1. All alphabetic characters in the strings in close proximity are part of one token; likewise with numbers.
2. Whitespace characters like space or line break or punctuation characters separate the tokens.
3. The resulting list of tokens may or may not contain punctuation and whitespace

2.4.1.2 Stop Word Removal

Sometimes a very common word, which would appear to be of little significance in helping to select text matching user's need, is completely excluded from the vocabulary. These words are called "stop words" and the technique is called "stop word removal". The general strategy for determining a "stop list" is to sort the terms by collection frequency and then to make the most frequently used terms, as a stop list, the members of which are discarded during indexing.
Some of the examples of stop-word are: a, an, the, and, are, as, at, be, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were. For instance: If operated on a token "saw", stemming may return just "s". While lemmatization will go through the morphological analysis and return see or saw depending upon the use of word "saw" as a verb or noun in the sentence(Lama, 2013).

2.4.1.3 Part of Speech Tagging

Part-of-Speech (POS) tagging means word class assignment to each token. Its input is given by the tokenized text. Taggers have to cope with unknown words (Out-Of-Vocabulary (OOV) problem) and ambiguous word-tag mappings. Rule-based approaches like ENGTWOL operate on;

1. Dictionaries containing word forms together with the associated POS labels and morphological and syntactic features and
2. Context sensitive rules to choose the appropriate labels during application.

2.4.1.4 Synonym Expansion

Synonym expansion, also known as lexical substitution, is the task of replacing a certain word in a given context with another suitable word similar in meaning. When a word has multiple meanings, synonym expansion tries to find the correct meaning of the word used in a sentence by identifying its synonyms (or substitutes) in a given context (Lama, 2013).

2.4.2 Text Processing

Text processing is the use of a data-mining algorithm to process the prepared data, compressing and transforming it to identify latent nuggets of information. At this stage, a fully featured NLP system would determine canonical and variant identities of entities identify conceptual relations between entities, and even instantiate particular frames of interest. Slot-filling of participants, dates and outcomes, as well as tables of extracted entities and relations, provides meaningful features for standard algorithms and techniques such as decision trees, neural networks, case-based learning, association rules or genetic algorithms (Lama, 2013).

2.4.2.1 Information Extraction

Information Extraction (IE) is an important process in the field of NLP in which factual structured data is obtained from an unstructured natural language document or text. Often this involves defining the general form of the information that we are interested in as one or more templates, which are then used to guide the further extraction process. IE systems rely heavily on the data generated by NLP systems (Lama, 2013). Tasks that IE systems can perform include:

Term analysis: This identifies one or more words called terms, appearing in the documents. This can be helpful in extracting information from the large documents like research papers which contain complex multi –word terms.

Named-entity recognition: This identifies the textual information in a document relating the names of people, places, organizations, products and so on.

Fact extraction: This identifies and extracts complex facts from documents .Such facts could be relationships between entities or events.

Two typical text mining techniques employed during this stage are classification and clustering.

2.4.2.2 Clustering

Clustering is an unsupervised pattern classification method that takes place during the text processing stage, whereby a group of ***n*** objects is classified into ***m*** partitions without prior knowledge whereby the number of partitions ***m*** may or may not be known initially. Clustering can be also be defined as the "the process of categorizing objects into groups whose members are similar in some way" (Algorithms, 2013).

The most widely researched and used clustering algorithm is the K-means algorithm that attempts to solve the clustering problem into a fixed number of clusters K known clusters in advance (Song and Park, 2009) . K-means classifies objects based on attributes or features into K number of groups. The algorithm is made up of the following steps:

1. Places K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assigns each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculates the positions of the K centroids.
4. Repeats Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

The K-means algorithm aims at minimizing an objective function, in this case a squared error function. The objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 ,$$

where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the *n* data points from their respective cluster centres (Algorithms, 2013).

2.4.2.3 Classification

Text classification is the automated assignment of natural language texts to predefined categories. Text classification is the primary requirement of text retrieval systems, which retrieve texts in response to a user query, and text understanding systems, which transform text in some way such as producing summaries, answering questions or extracting data. Existing supervised learning algorithms to automatically classify text need sufficient documents to learn accurately Figure 6 demonstrates the text classification process that begins from text pre-processing (Ikonomakis et al., 2005).The most common techniques used for automatic text classification include; Association rule mining, implementation of Naïve Bayes classifier, genetic algorithm, decision tree (Kamruzzaman et al., 2010) and K Nearest Neighbor.

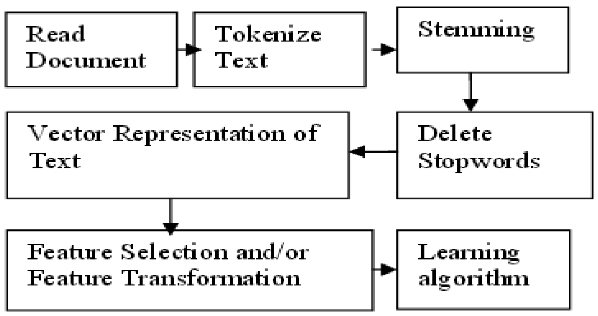

Read Document
Tokenize Text
Stemming
Delete Stopwords
Vector Representation of Text
Feature Selection and/or Feature Transformation
Learning algorithm

Figure 6: Text Classification Process

Association rule mining

It finds interesting association or correlation relationships among a large set of data items, the association rule is based on associated relationships. The discovery of interesting association relationships among huge amounts of transaction records can help in many decision-making processes. Association rules are generated on the basis of two important terms namely minimum support threshold and minimum confidence threshold (Kamruzzaman et al., 2010). Association rule mining is a two-step process, which includes:

1. Find all frequent itemsets(set of items).
2. Generating strong association rules from the frequent itemsets.

The Naïve Bayes (NB) classifier

A Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions(Ramesh and Ramar, 2011). It assumes that the occurrence of each word in a document is conditionally independent of all other words in that document given its class. Bayesian classifiers have exhibited high accuracy and speed when applied to large database. While applying NB classifier to classify text, each word position in a document is defined as an attribute and the value of that attribute to be the word found in that position. Here NB classification can be given by:

$$V_{NB} = \text{argmax } P(V_j)\ \Pi\ P(a_j \mid V_j)$$

Here V_{NB} is the classification that maximizes the probability of observing the words that were actually found in the example documents, subject to the usual N independence assumption. The first term can be estimated based on the fraction of each class in the training data. The following equation is used for estimating the second term:

$$\frac{n_k + 1}{n + \mid \text{vocubulary} \mid}$$

where n is the total number of word positions in all training examples whose target value is V_j, n_k is the number of items that word is found among these n word positions, and | vocubulary | is the total number of distinct words found within the training data (Kamruzzaman et al., 2010).

Bayesian classifiers are popularly used not only for text categorization, but also for any other classification problems, since their learning is fast and simple. Despite their naive design and apparently over-simplified assumptions, NB classifiers often work much better in many complex real-world situations than expected. An advantage of the NB classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix (Ramesh and Ramar, 2011).

Genetic algorithm

A genetic Algorithm starts with an initial population which is created consisting of randomly generated rules. Each rule can be represented by a string of bits. Based on the notion of survival of the fittest, a new population is formed to consist of the fittest rules in the current population, as well as offspring of these rules. Typically, the fitness of a rule is assessed by its classification accuracy on a set of training examples. In general, genetic algorithm starts with an initial population which is created consisting of randomly generated rules. Each rule can be represented by a string of bits.

As a sample example, suppose the samples in a given training set are described by two Boolean attributes, A1 and A2, and that there are two classes, C1 and C2.The rule "IF A1 AND NOT A2 THEN C2" can be encoded as the bit string "100", where the two leftmost bits represent attributes A1 and A2, respectively and the rightmost bit represents the class. Similarly, the rule "IF NOT A1 AND NOT A2 THEN C1" can be encoded as the bit string "001", If an attribute has k-values, where k>2, then k-bits may be used to encode the attribute's values. Classes can be encoded in a similar fashion. Based on the notion of survival of the fittest, a new population is formed to consist of the fittest rules in the current population, as well as offspring of these rules. Typically, the fitness of a rule is assessed by its classification accuracy on a set of training examples. Offspring are created by applying genetic operators such as crossover and mutation. In crossover, substrings from pairs of the rules are swapped to form new pairs of rules. In mutation, randomly selected bits in a rule's string are inverted. The process of generating new populations based on prior populations of rules continues until a population P "evolves" where each rule in P satisfies a pre-specified fitness threshold (Kamruzzaman et al., 2010).

Decision trees

A decision tree model consists of internal node and leaves. Each of the internal node has a decision associated with it and each of the leaves has a class label attached to it. A decision tree based classification consists of two steps.

1. Tree induction whereby tree is induced from the given training set.
2. Tree pruning: The induced tree is made more concise and robust by removing any statistical dependencies on the specific training data set (Ghosh et al., 2012).

The decision tree classification method is outstanding from other decision support tools with several advantages like its simplicity in understanding and interpreting, even for non-expert users. However decision trees are built by greedy search algorithms, guided by some heuristic that measures "impurity". Irrelevant attributes may affect badly the construction of a decision tree. Small variations in the data can imply that very different looking trees are generated (Patra and Singh, 2013).

K-Nearest Neighbor (KNN)

KNN classifier is a classification algorithm where objects are classified by voting several labeled training examples with their smallest distance from each object. The major disadvantage of KNN is that it uses all features in computing distance and costs very much time for classifying objects (Patra and Singh, 2013).

Support vector machines (SVM)

It is a method for classification of linear and non-linear data. This algorithm uses non-linear mapping to transform training data into higher dimension and then it search for linear optimal separating hyper plane. SVM optimizes the weights of the inner products of training examples and its input vector called Lagrange multipliers, instead of those of its input vector, itself, as its learning process it provides a compact description of the learned model (Patra and Singh, 2013).

The advantages of SVM are; effective in high dimensional spaces, still effective in cases where number of dimensions is greater than the number of samples, uses a subset of training points in the decision function (called support vectors), so it is also memory efficient and versatile as different kernel functions can be specified for the decision function.

Latent semantic indexing (LSI)

This also known as latent semantic analysis (LSA), is an indexing and retrieval method that uses singular value decomposition (SVD) to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text. LSI is based on the principle that words that are used in the same contexts tend to have similar meanings. A key feature of LSI is its ability to extract the conceptual content of a body of text by establishing associations between those terms that occur in similar contexts ("Latent semantic indexing," 2014).

How it works

It decomposes a term-document or text into a product of matrices using SVD.

$$\text{SVD: } C = U\Sigma V^T \text{ (where } C = \text{term-document matrix)}$$

SVD is used to compute a new and improved term-document matrix c' which is then used to get better similarity values (Gray, 2011). Once a term-document matrix is constructed, local and global weighting functions can be applied to it to condition the data. The weighting functions transform each cell, a_{ij} of A, to be the product of a local term weight, l_{ij}, which describes the relative frequency of a term in a document, and a global weight, g_i, which describes the relative frequency of the term within the entire collection of documents ("Latent semantic indexing," 2014).

Why use LSI

1) It has been shown to provide superior performance to other information retrieval techniques in a number of controlled tests. 2) A number of experiments have demonstrated a remarkable similarity between LSI and the fundamental aspects of the human processing of language. 3) It is immune to the nuances of the language being categorized, thereby facilitating the rapid construction of multilingual categorization systems. 4) It can perform well with very limited quantities of training data, generally with only a few examples per category (Zukas and Price, 2003).

2.4.3 Text Analysis

Text analysis is the evaluation of the output to see if knowledge was discovered and to determine its importance. Having run the algorithms, the mined text is submitted to various techniques that will enable direct usage of the mined information, either by a Link Discovery tool or by visualization in a tool that will enable human analysts to complete the analysis begun by the text mining technology (Liddy, 2000).

2.5 Synthesis and Approach

Table 2 shows a summary of techniques reviewed and the identified gaps;

Table 2: Review of Techniques Studied

Methodology	**Critique**
CBR	1. May require large storage space requirements and large processing 2. Cases may need to be created by hand 3. Needs case-base, case selection algorithm, and possibly case-adaptation algorithm. Systems developed using CBR are by customer support personnel to determine possible solutions.
CA	1. Require sufficient language processing capabilities. 2. Must operate effectively in the enterprise; scalable and reliable, and must integrate cleanly into existing business processes.
Text classification Algorithms	
NB	Over-simplified assumptions that may not be applicable in the real world.
Decision Trees	1. Irrelevant attributes may affect badly construction of a decision tree. 2. Small variations in the data can imply that very different looking trees are generated
KNN	1. It uses all features in computing distance 2. Costs very much time for classifying objects
SVM	1. Training speed is low 2. If features are much greater than samples likely to perform poorly. 3. Do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

The study implemented automated user support using conversational agents conversed with customers in a structured way. To address the NLP requirements the study used NB classification and LSI algorithm to categorize customer requests to determine the appropriate response to give customer and what actions to perform.

3 METHODOLOGY

3.1 Introduction

The chapter covers the system development methodologies that were used in the study.

3.2 System Development Methodology

3.2.1 Agile Software Development Methodology

This methodology was employed for the development of the administration web interface, automatic email saving component and data generation and saving components. The methodology puts extreme emphasis on delivering working code or product while downplaying the importance of formal processes and comprehensive documentation. Proponents of these methodologies argue that by putting more emphasis on "actual working code," software development processes can adapt and react promptly to changes and demands imposed by their volatile development environment (Kruchten, 2001). Figure 7 illustrates a generic agile development process that features an initial planning stage, rapid repeats of the iteration stage, and some form of consolidation before release.

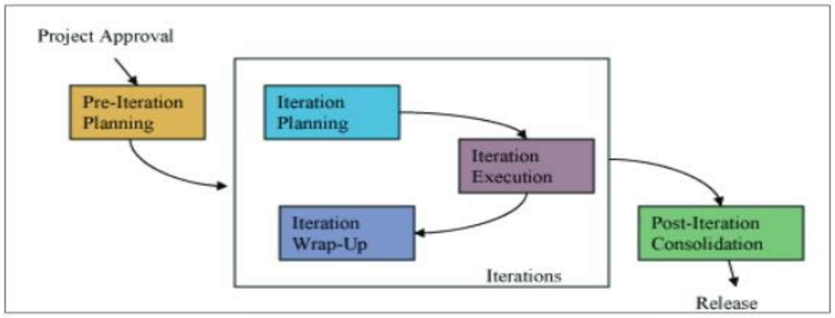


Figure 7: A Generic Agile Development Process

3.2.2 *Multi-Agent Systems Engineering Methodology*

The methodology used in development of the conversational agents was Multi-Agent Systems Engineering (MaSE) methodology. MaSE builds on object-oriented techniques and uses defined in the Unified Modeling Language (UML) (Wood and DeLoach, 2001). MaSE was used to analyze, design, and implement the conversational agents by proceeding in an orderly fashion through the development lifecycle.

MaSE as illustrated by Figure 8 consists of two phases' analysis and design. The analysis phase involves a) Capturing goals, b) Applying use case 3) Refining roles and results in the models goal hierarchy, use cases, sequence diagrams, concurrent tasks, role model. The Design Phase involves a) Creating agent classes, b) Constructing conversations, c) Assembling agent classes and d) System design and results in creation of agent class diagrams, conversation diagrams , agent architecture diagrams and deployment diagrams.

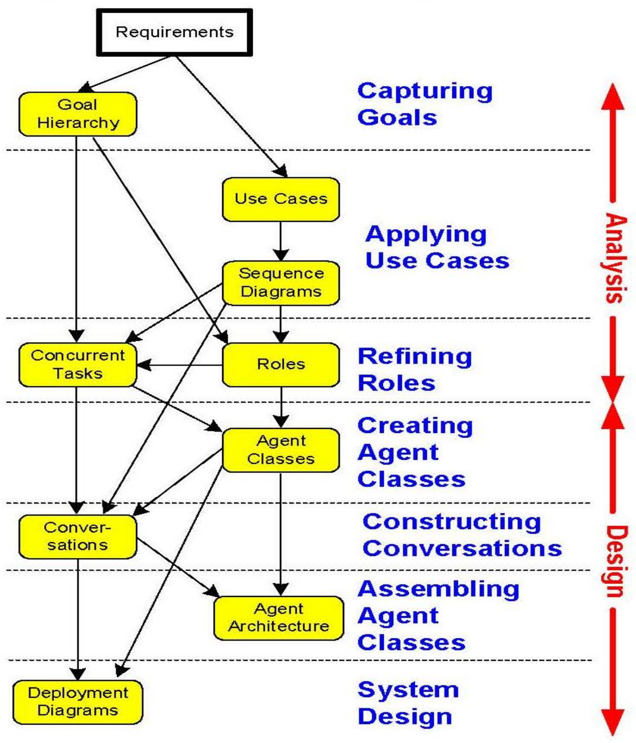


Figure 8: MaSE Development Stages

3.3 Data Sources

The study used the University Online application user support emails as its core source of data for support requests and responses, sample data can be viewed in Appendix A. Data on applicants was retrieved from institutional websites and generated randomly.

3.4 Data Collection

The data was provided as series of forwarded emails from the official online application support email account. The emails were downloaded then read and saved automatically into the database. The emails were then used to create training data for the classifier and to validate the prototype.

Other data was manually captured from institutional websites using data entry forms.

3.5 General System Architecture

Figure 9 illustrates the general system architecture.

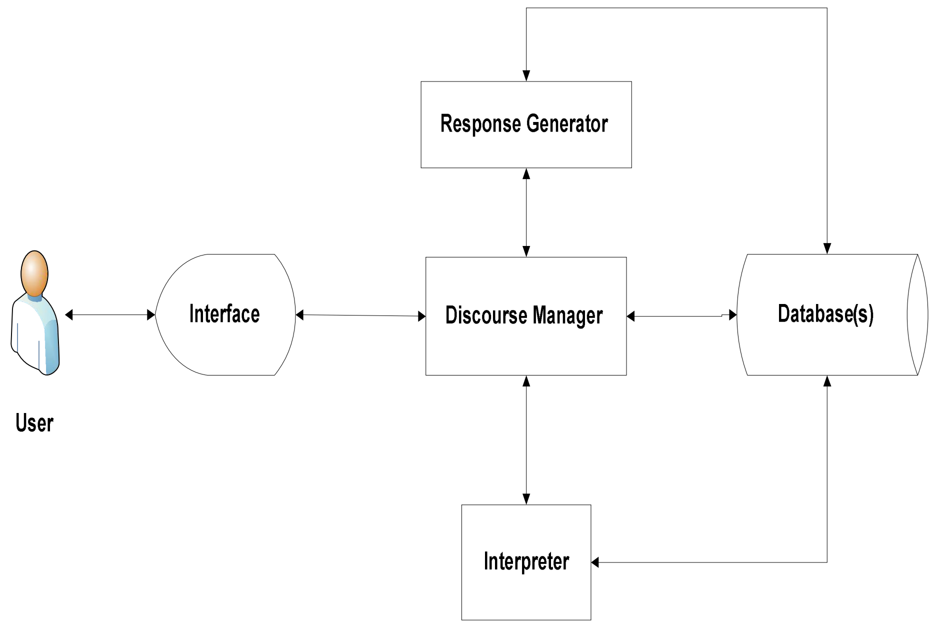


Figure 9: General System Architecture

The following are automated support system components are;

Interface

What a customer uses to relay their requests and what is used by the system to communicate any information to the customer. In this study chosen interface is email.

Discourse Manager

The discourse manager is the core of the system it controls the dialogue between the customers and the automated customer support system. If it will perform the following functions;

Interpreter

This is the Natural language understanding (NLU) unit which utilizes NB or LSI text mining algorithm to determine the meaning of text which it will then pass to the discourse manager. The interpreter performs text processing as described in the literature review to classify the requests into various categories. Once the text has been categorized it will store the uncategorized text and the categorized text in created databases.

Response Generator

This receives the classified text and determines the appropriate action to be performed or message to be sent to the customer. It consists of a set of production rules used to determine the correct actions or responses to give depending on the category the email has been placed.

3.6 Prototype Validation and Comparative Analysis

Figure 10 illustrates the general architecture of the comparative analysis component of the system.

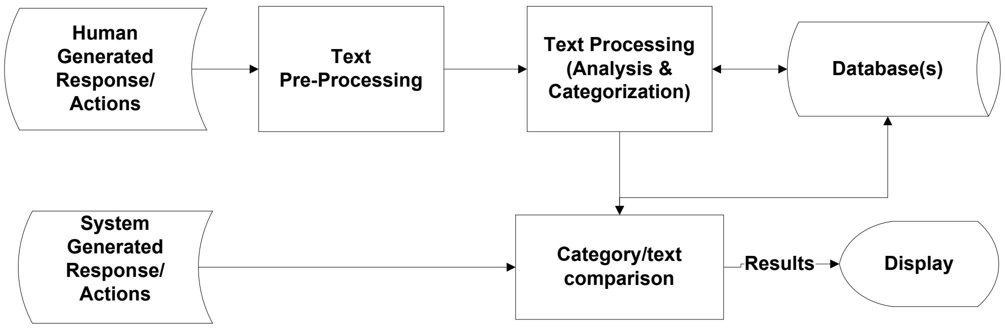

Human Generated Response/ Actions
Text Pre-Processing
Text Processing (Analysis & Categorization)
Database(s)
System Generated Response/ Actions
Category/text comparison
Results
Display

Figure 10: Comparative Analysis Design

3.7 Study Activities

Table 3 shows a summary of activities that were carried out to realize the objectives of this study.

Table 3: Study Activities

	Objective	**How the objective was achieved**
1.	To identify and analyze techniques used in automating customer support and select suitable technique for creating a customized automatic customer support system.	Conducted desk research on techniques used to automate customer support. Analyzed all the techniques identified and reviewed their advantages and disadvantages and selected the most suitable for the study.
2.	To develop a customized automatic customer support system.	Implemented the identified method through development of prototype.
3.	Conduct a comparative analysis of the performance of the system developed with human customer support.	Analyzed the responses generated by the automated customer support system and compared with those generated by customer support personnel to identify the level of accuracy of the developed system.

4 ANALYSIS, DESIGN AND IMPLEMENTATION

4.1 Introduction

This chapter covers the system's requirements, the design and finally the implementation of system.

4.2 Systems Analysis

4.2.1 Functional Requirements

The following are the functional requirements for the prototype components;

Interface

1. Should be able to send and receive support emails from and to various email accounts.
2. Should allow polling by a local application to retrieve new emails.

Discourse Manager

The discourse manager is the core of the system it will control the dialog between the customers and the customer support system. It should be able to;

1. Poll and retrieve emails from the support email account.
2. Ignore duplicate or empty emails and save valid new emails only.
3. Perform text pre-processing on retrieved emails before passing the text to the interpreter for analysis. Pre-processing should involve removing non ascii characters, removal of html tags, stripping text of excess spaces and removal of stop words.
4. Interact with the interpreter and the response generator to; receive and save classifications, relay classifications to response generator and store responses.
5. Relay messages to the customer via the email account.

Interpreter

The interpreter should do the following;

1. Categorize the emails received

2. If it is unable to determine a category it should relay this to the discourse manager for escalation to an administrator

Response Generator

It will should be able to do the following based on the category text has been placed in;

1. Determine what email response to send to a customer and action to perform.
2. It should be able to relay response and action to the dialogue manager.

Admin interface

The interface should do the following;

1. Authenticate users based on username and password set to allow access.
2. Allow start, stop and viewing of current run status of the application.
3. Allow change of application configurations.
4. Allow addition, update or delete of data from the support and applicant databases.
5. Allow the view of the system logs as the application is run.

4.2.2 Non-Functional Requirements

A non-functional requirement specifies systems' properties and constraints. The following are the non-functional requirements for the application;

1. Administrator's graphical user interface should be easy to use and navigate.
2. Performance requirements
    a. Training time for the classifier should be low.
    b. The system should have short response times for emails once received.
3. Operating constraints: The application should not exhaust available resources that is; memory and processor by creation of agents, there should be a mechanism to limit agents created.

   To run the application a server requires to meet the following minimum system requirements; RAM 1GB, CPU 2.4 GHz and 64 bit system

4. Platform constraints: The application should run operate in either Windows or Linux environment but there are noted difficulties involved in optimizing the classifier in Windows.

5. Accuracy and Precision: The classifier should be as accurate as possible even with minimal training data.

4.3 System Design

4.3.1 Architectural System Design

The Figure 11 illustrates the detailed architectural design of the system.

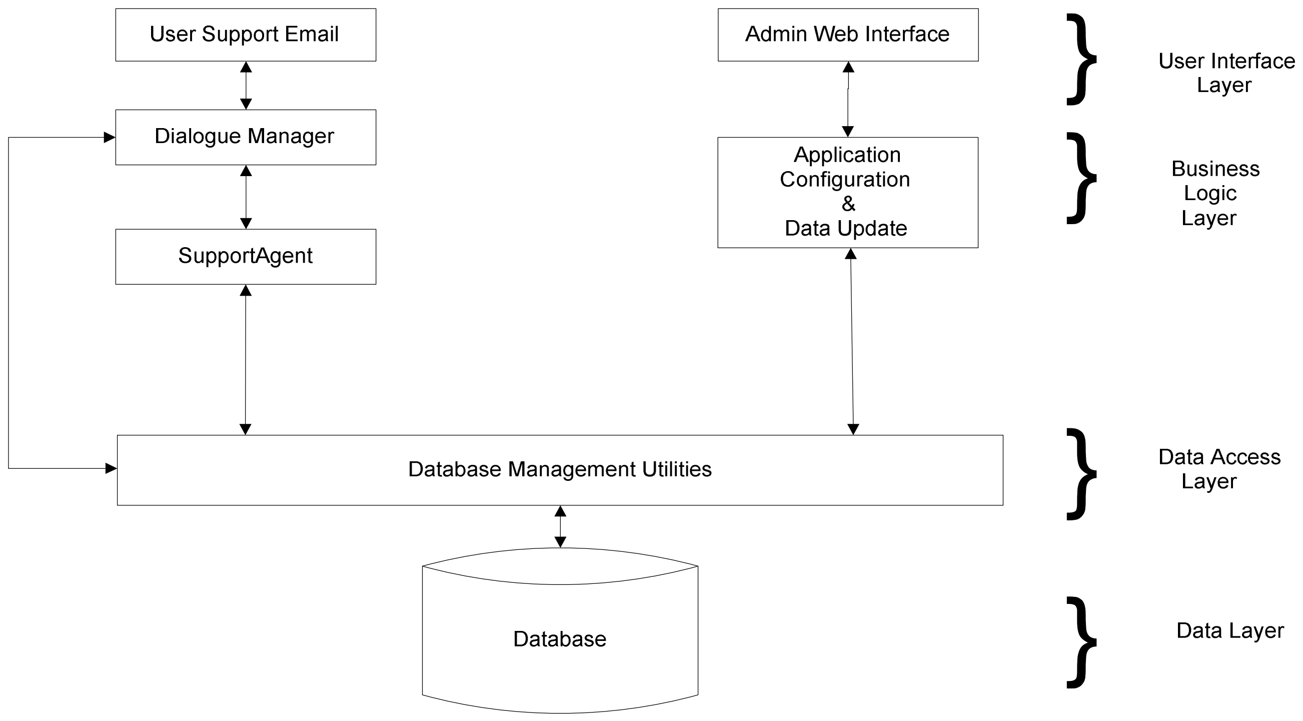


Figure 11: Detailed System Architecture

1. The **User Interface Layer** enables users to interact with the system. The interface for customers is email while that for administrators is web pages. The interface layer will carry out user commands and presents results generated by the system to the user.
2. The **Business Logic Layer** is the as main system engine. It has the working logic for the system; which it includes the dialogue manager, support agent and the administrative backend processing.

3. The **Data Access Layer** serves retrieval requests from the upper layers these include; data storage, data retrieval and data crosschecking.
4. The **Data Layer** this is the data store for all data pertaining to the application, includes support related data and applicant data.

4.3.2 Conversational Agent Design

Each conversational agent houses the following components explicitly or implicitly;

1. Part of the discourse manager
2. Interpreter
3. Response generator

The methodology used in design of the conversational agent is MaSE which entailed two phases; analysis and design that yielded the models highlighted in the following sections. The agents designed and developed are purely reactive, responding to change in their internal states.

Capturing Goals

This phase takes the initial system specification and transforms it into a structured set of system goals as shown in a Goal Hierarchy Diagram (Figure 12). The Goal Hierarchy Diagram is a directed, acyclic graph where the nodes represent goals and the arcs define a sub-goal relationship.

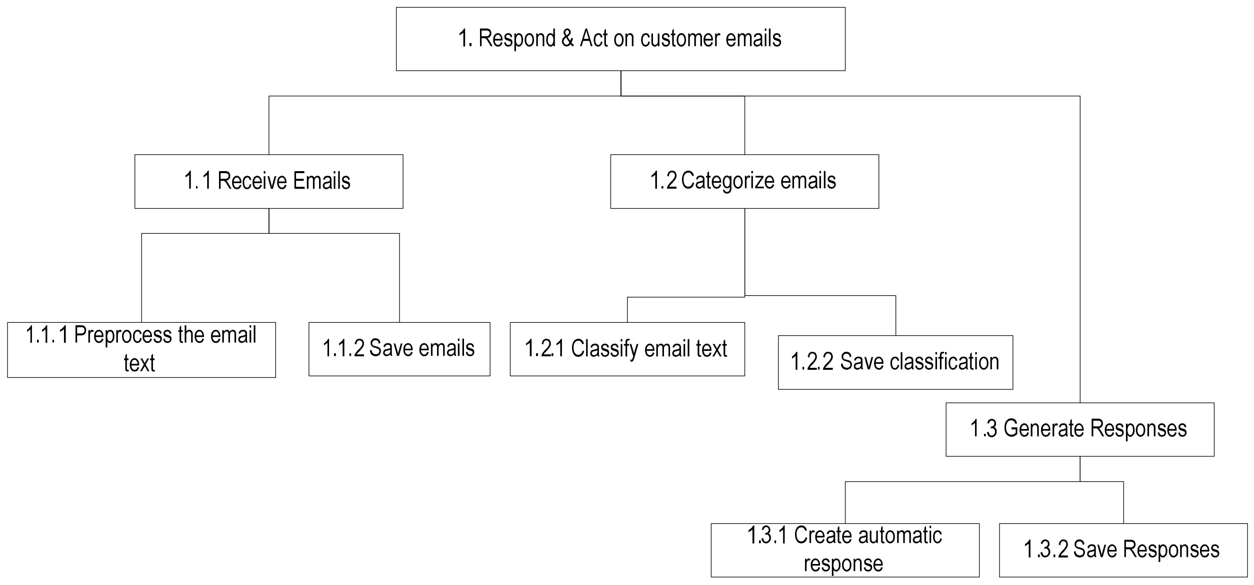

1. Respond & Act on customer emails
1.1 Receive Emails
1.2 Categorize emails
1.1.1 Preprocess the email text
1.1.2 Save emails
1.2.1 Classify email text
1.2.2 Save classification
1.3 Generate Responses
1.3.1 Create automatic response
1.3.2 Save Responses

Figure 12: Goal Hierarchy Diagram

Applying Use Cases

This phase captures use cases from the initial system requirements and restructures them as a sequence diagram (Figure 13), which depicts a sequence of messages between the agent roles.

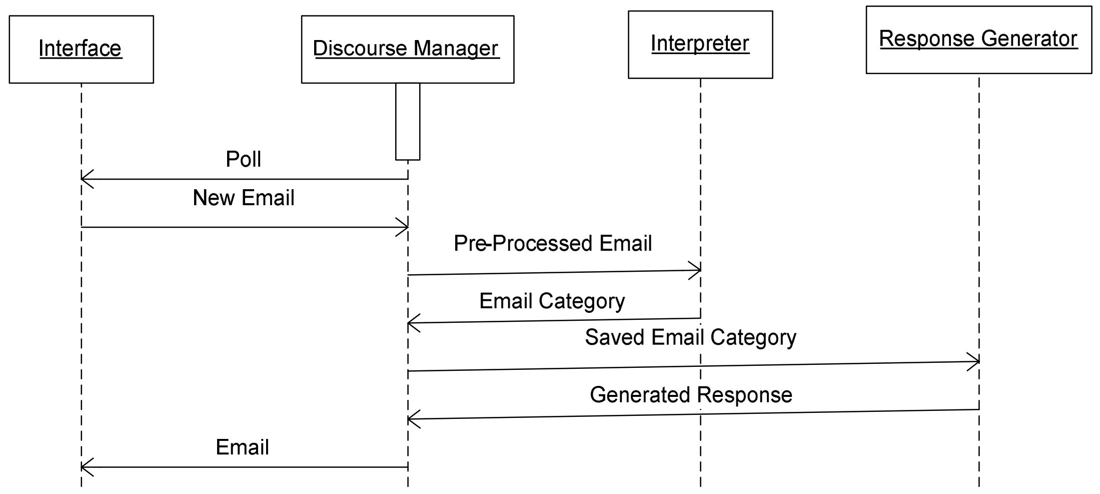


Figure 13: Sequence Diagram

Creating Agent Classes

The product of this step of the Design phase is an agent class diagram (Figure 14), that depicts the agent classes and the conversations between them.

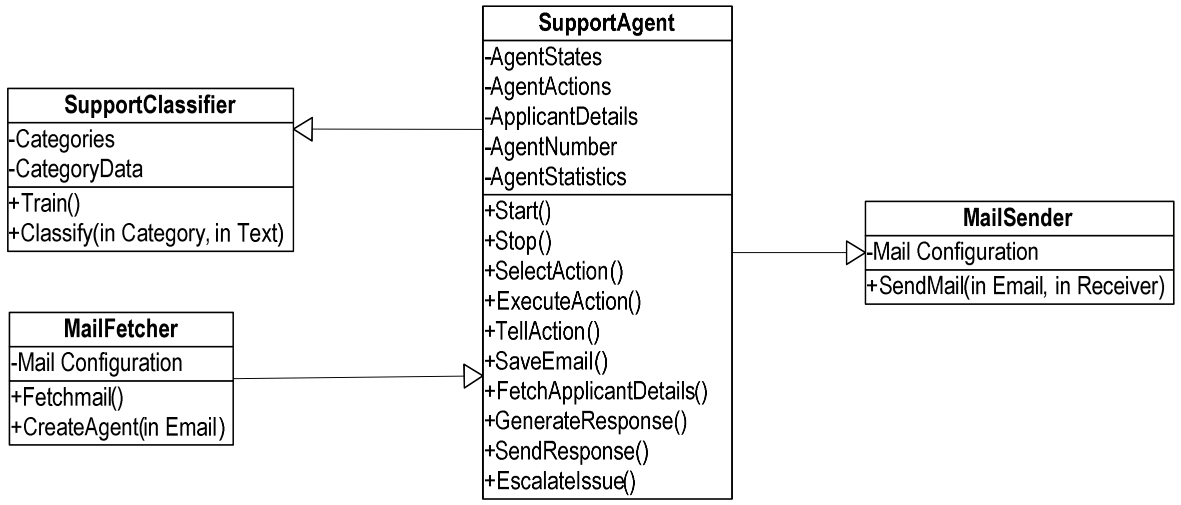


Figure 14: Class Diagram

4.3.3 Classifier Design

The interpreter component of the system was designed to use two classifier algorithms NB and LSI.

4.3.3.1 Naïves Bayes Classifier Design

While applying NB classifier to classify text, each word position in a document is defined as an attribute and the value of that attribute to be the word found in that position. The NB classification is given by:

$$V_{NB} = \text{argmax } P(V_j) \, \Pi \, P(a_j \mid V_j)$$

Here V_{NB} is the classification that maximizes the probability of observing the words that were actually found in the example documents, subject to the usual N independence assumption.

Pseudo code

The following section shows the pseudo code for the NB classifier used. The **Train** function returns; **CategoriesCount** array which holds the categories and the number of training data for each, **Categories** array which holds the category, words in training data and the number of times each word appears per category and the **Totalwords** this is a counter of all the words used in training. The **Classifications** function is called by the classify function and it returns the **Score** array which holds the category and a score associated with the category. The **Score** array is ordered downward and the category with the largest value is that which is assigned to test. In the event that two or more categories have the same score then the text is not classified in any category.

```
CategoriesCount = [] #Initialize category counts array
Categories = [] #Initialize categories array
Totalwords = [] #Initialize total words counter
Train(category,text) #The train function will receive categories and text from the database
{
    1. CategoriesCount[category] +=1   #Increment the counter for this category
    2. WordCount = Hasher.Word_hash(text) # Create an array that holds the words and the number of
       times they appear in the text
    3. For Each WordCount as word => count   # Loop through the array
    4. Categories[category][word] += count  # increment  counter for the word with the count
    5. Totalwords += count   # Increment the counter for total words with the word count
    6. End For Each #end the loop
}
```

```
Classifications(text){
    1.  Score =[] #initialize the scores array
    2.  For Each CategoryCount as k => catcount  #Loop through the array
    3.  training_count +=catcount #Increment with the number of training data for each category
    4.  End For Each #End loop
    5.  For Each Categories as category=> category_words #Loop through the array created during
        training
    6.  Score[category]=0 #initialize the score for the category
    7.  For Each category_words as k => catcount #Loop through the array of words for each category
    8.  total  += catcount #increment the variable with the number of words in each category
    9.  End For Each
    7.  WordCount = Hasher.Word_hash(text) ) # Create an array that holds the words and the number of
        times they appear in the text
    10. For Each WordCount as word=> count # Loop through the array
    11. If category_words has key word or category_words[word] exists
    12. s=0.1
    13. Score[category] = log (s/total)
    14. End If
    15. End For Each
        # Adding prior probability for the category
    16. If category_words has key word or category_words[word] exists
    17. s=0.1
    18. Score[category] = log (s/ training_count)
    19. End If
    20. End For Each
    21. return Score
}
Word_hash(text)
{
    1.  Remove extra punctuation, short symbols and tags
    2.  Words = text.split # Split the text to create and array of words
    3.  d =[] #initialize word count array
    4.  d_s =[] #initialize synonym count array
    5.  For Each Words as word #Loop through the array
    6.  Set the word to lowercase
    7.  If word is not contained in CORPUS_SKIP_WORDS and if it is length greater than 2
    8.  synonym =stem(word) #Find synonyms for the word
    9.  d_s[synonym] += 1 #increment the array counter for the synonym
    10. End If
    11. d[word] += 1 #increment the array counter for the word
    12. End For Each
    13. return d + d_s #merge the arrays for synonyms and for words
}
```

```
IssueClassifier(issuetext, issue_id)
{
    1. classificationArray = Classifications(issuetext) #Classify the text
    2. Sort the classifications weights returned in ClassificationsArray in Descending order
    3. proposedcatweight = classificationArray[0][1] #Get the highest score
    4. samecount =0 #Initialize the counter for checking the number of categories with the same weight
    5. For Each classificationArray as k => value #loop through the array of categories & scores
    6. If proposedcatweight==value #compare the returned score with those returned
    7. Samecount +=1 #increment the counter if the category has the same score returned
    8. End If
    9. End For Each
    10. If issue_id is not empty #Check if the issue_id has been received
    11. Save to classifications table all the weights #If it has save to the database the categories & scores
    12. End If
    13. If samecount >=2  # If the counter has value greater than or equal to 2
    14. classification ="" #Set no classification
    15. else
    16. classification = classificationArray[0][0] #Return the category with the highest weight
    17. End If
    18. return classification
}
```

4.3.3.2 Latent Semantic Indexing Classifier Design

How it works

1. It decomposes a term-document or text into a product of matrices using SVD.

$$\text{SVD: } C = U\Sigma V^T \text{ (where } C = \text{term-document matrix)}$$

2. SVD then is used to compute a new and improved term-document matrix c' which is then used to get better similarity values (Gray, 2011).
3. Once a term-document matrix is constructed, local and global weighting functions can be applied to it to condition the data. The weighting functions transform each cell, a_{ij} of A, to be the product of a local term weight, l_{ij}, which describes the relative frequency of a term in a document, and a global weight, g_i, which describes the relative frequency of the term within the entire collection of documents ("Latent semantic indexing," 2014).

Pseudo code

The following section indicates the pseudo code for the LSI classify used. The same **Word_hash** function is used as in the NB Classifier.

```
Items = [] #Initialize category counts array
Train(category, text)
 {
    1. text = clean(text)Remove extra punctuation, short symbols and tags
    2. Items[text]= [text,categories]#load the cleaned text into an array
    3. Word_list = Word_hash(text) # Create an array that holds the words and the number of times they
       appear in the text
    4. Create an index vector of all words and the number of times they appear in the text using the
       Word_list array
 #No. 4  will decompose the text into term-document array using SVD  u, v, s = matrix.SV_decomp

 }

classifycustom( text )
 {
    1. Use the vector created in Train function to assign votes for each category based on the  text
    2. Return the Votes array that shows the categories and the vote assigned per category for the text
       indicated
 }
Issueclassifier(issuetext, issue_id)
 {
      1. Votes = classifycustom( issuetext) # Get the votes array for the issue text
      2. If the Votes array is not empty
      3. Save the votes and categories to the classifications table
      4. Get the category with the highest weight as the classified category
      5. End If
      6. Return the category assigned
  }
```

The **word_hash** function removes the words that are defined in Appendix C section 9.1.

4.3.4 Logic/Data flow of the Application

Application Initialization

Figure 15 illustrates the logic that pertains to initialization of the application.

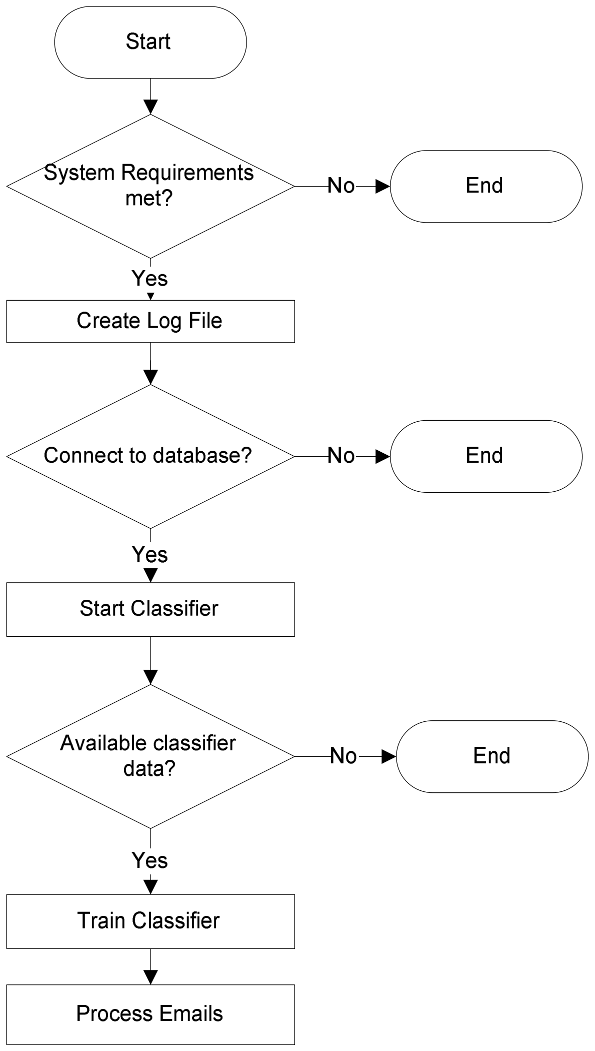


Figure 15: Application Initialization

The system requirements that are required to be met for application to run include installation and initialization of apache, mysql and ruby.

Application Logic

Figure 16 shows the data/logic flow during email processing; the steps followed during a typical run of the application. The figures also shows when the conversational agent is initialized, what functions it performs and finally when it is stopped.

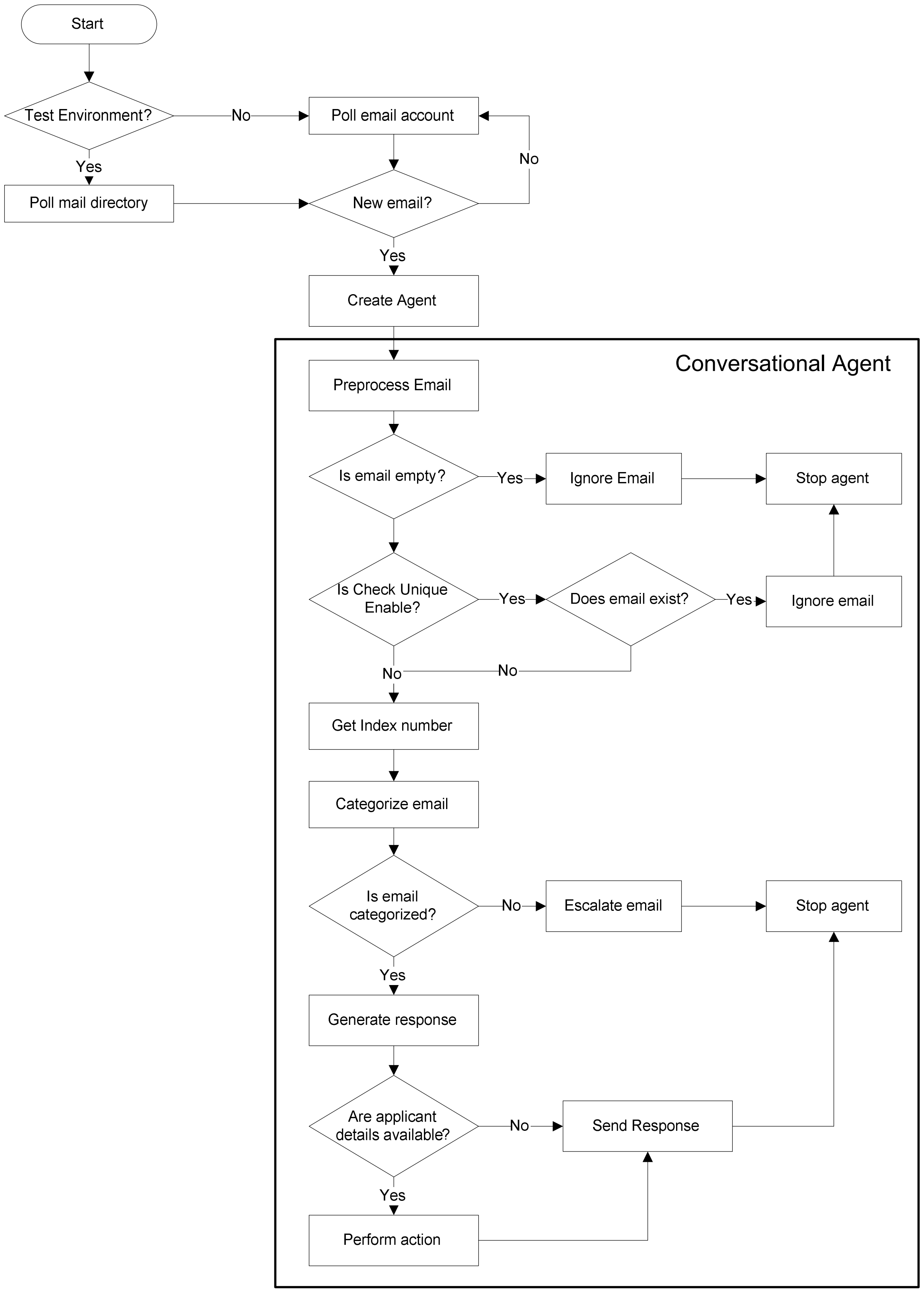


Figure 16: Application Data Flow

Figure 17 shows the logic flow during the retrieval of an applicant's details based on the index number numerical string provided in the email text. .

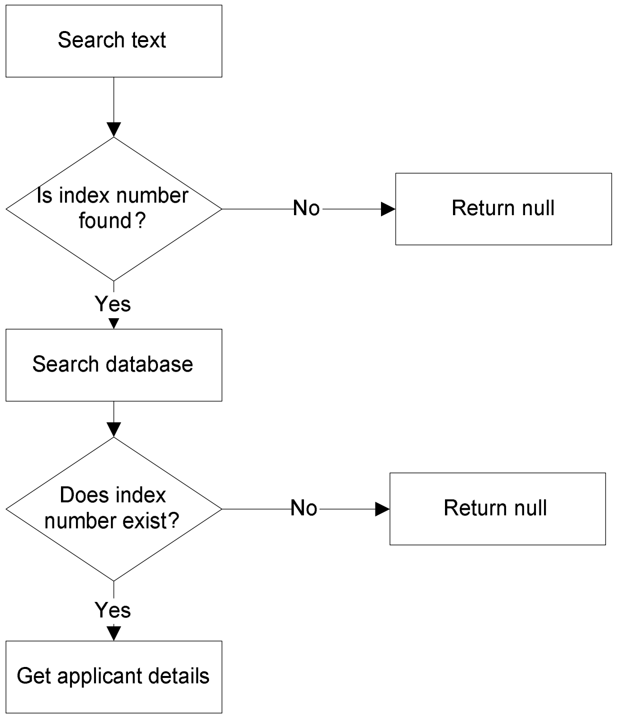


Figure 17: Applicant Details Fetch Data Flow

4.3.5 The Database Model

4.3.5.1 Support Database

Figure 18 shows the database model for the automatic customer support application. The tables shown store the following information;

1. application_run: Statistics of each run of the application, this includes; the number of emails received, agents created, emails ignored, escalated, responded and pending. It also shows the run time of the application.
2. class: The group which each category falls under
3. categories; The sub-groups which each class holds
4. categories_data: Data that is used by to train the classifier which is related to the categories table
5. support_issue: Emails received by polling an email account or a mail directory. The table holds the raw email received as well as the processed email.
6. issue_categories : Categories of each of the emails received

7. classifications: Weights that are generated per category when each classifier is used
8. issue_responses: Messages and actions that generated and sent per issue received
9. check_emails: Emails that are used to validate the application and to compare against the responses created by support staff
10. emails: support emails that have been automatically saved for creation of training data and for comparison purposes.
11. response_categories; The categories for the responses that will be sent out
12. response_categories_data: Data that is used by to train the response classifier that is related to the response_categories table

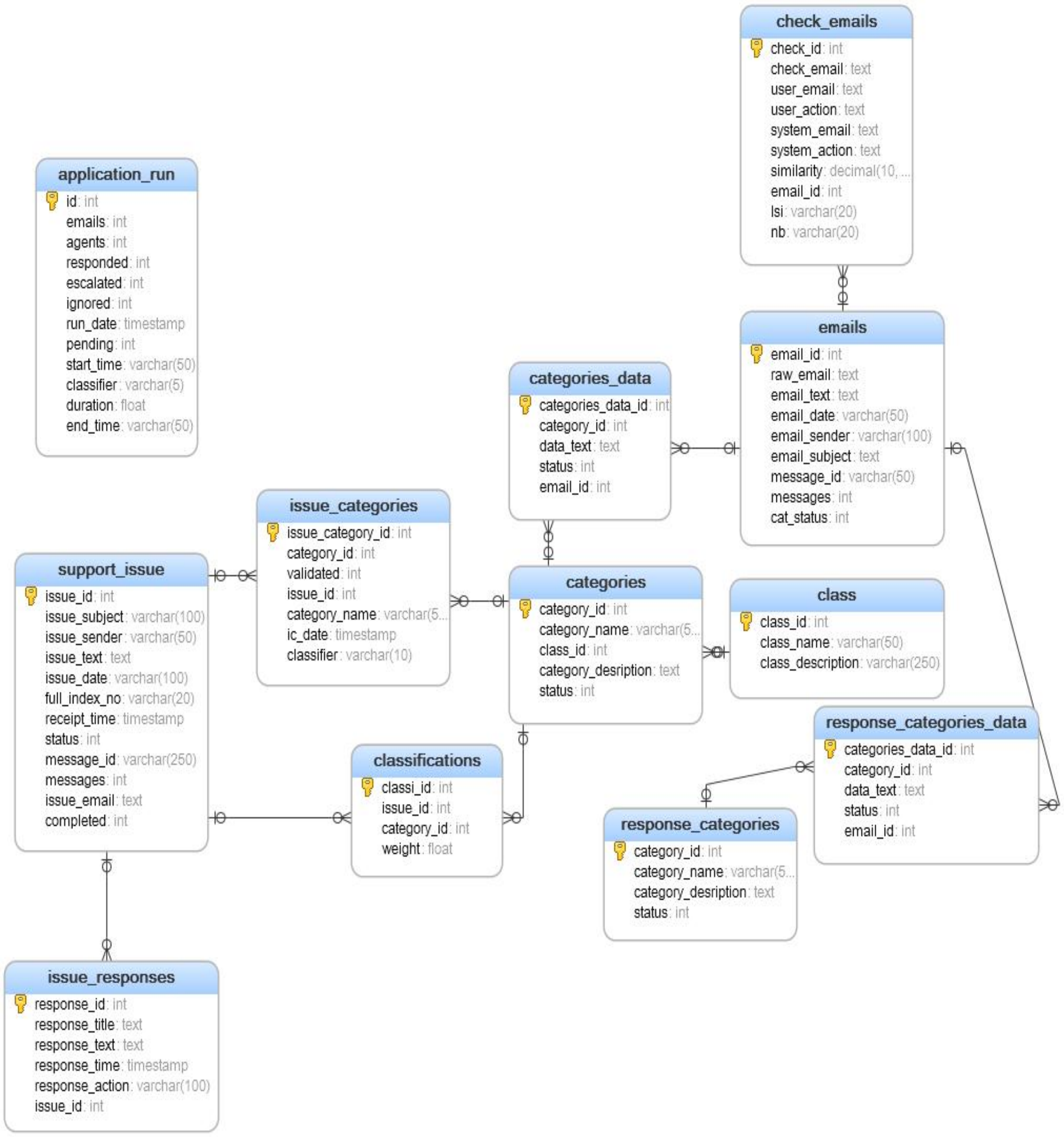


Figure 18: Support Database Model

4.3.5.2 Applicant's Database

Figure 19 shows the applicant's database model.

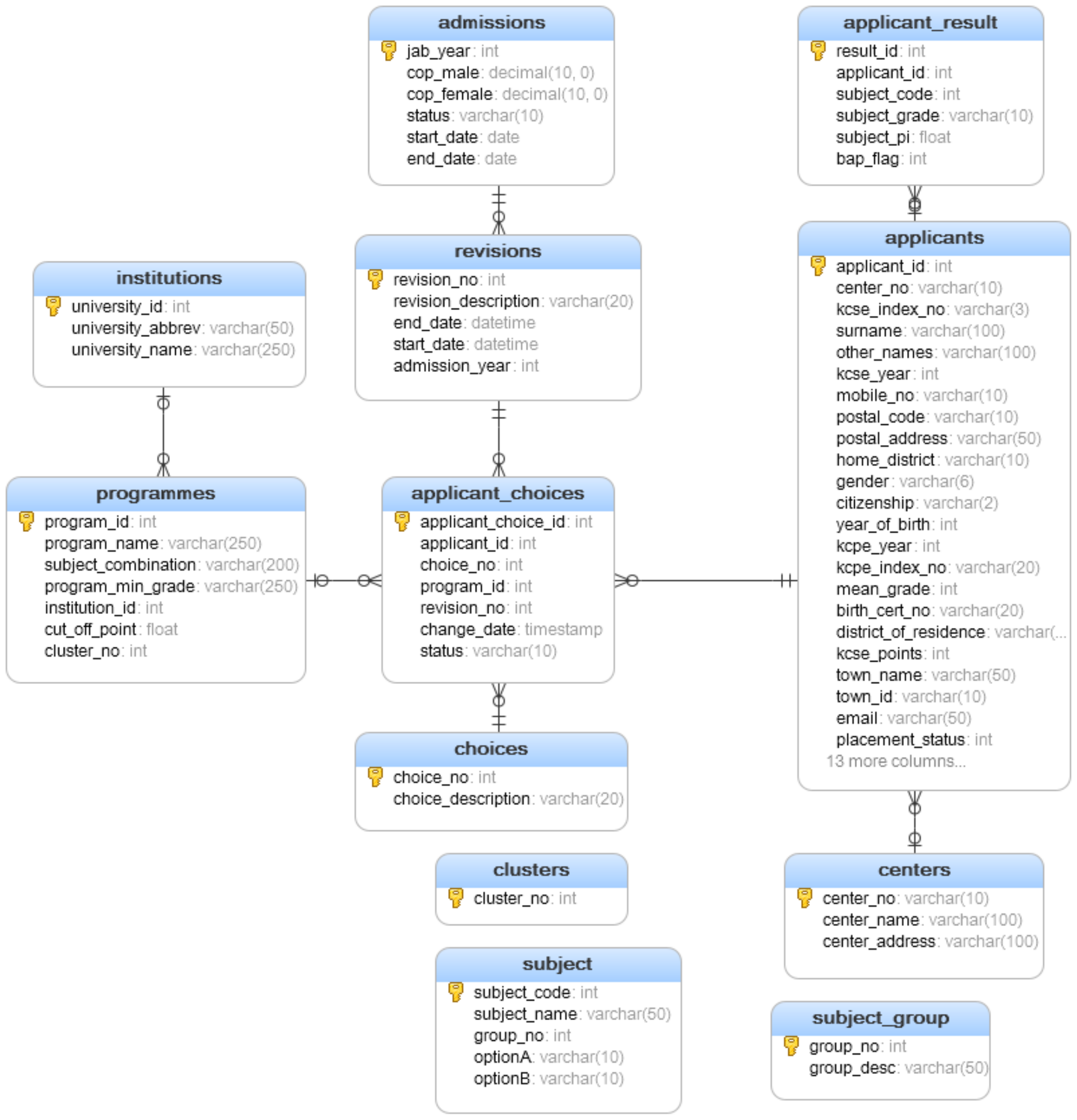


Figure 19: Applicants Database Model

The applicant's database consists of the tables illustrated above which hold the data;

1. admissions : Year of admission, the cut off points for male and females and the revision dates.
2. applicants: Applicant information that is names, index numbers and other details required for registration, processing and validation.
3. choices: Possible choices descriptions

4. applicant_choices: Programme choices made by applicants
5. applicant_result : Applicant results that have been sent by the examination body
6. centers: Examination centers that were used by applicants
7. clusters: List of all possible clusters
8. institutions : List of institutions which have registered with KUCCPS
9. programmes: List of programmes offered per institution
10. revisions: Possible revision dates

4.4 System Implementation

4.4.1 The Front End

The front end of the system was developed using PHP/HTML and MYSQL database. The PHP script interacts with the backend automatic support application Ruby service.

4.4.2 The Application Logic

The application was implemented in Ruby and MYSQL in Windows environment. It was also customized for Linux environment.

4.4.2.1 Email Retrieval

The email retrieval component was implemented using the Mail and Mailman libraries, and the email address used for implementation is a Gmail account, specifically p58.75763.2012.2@gmail.com.

4.4.2.2 Conversational Agents

The conversational agents were implemented using the AgentDispatcher multi-agent libraries more specifically the SimpleDispatcher class that extends the AgentDispatcher. Multithreading was used to enable parallel execution of multiple agents.

4.4.2.3 Classifier Development

The NB and LSI classifiers were implemented using the Classifier-Reborn library. To speed up the training of the LSI classifier the GNU Scientific Library (GSL) a numerical library was installed in the Linux environment. The training data used by the classifiers was loaded from MYSQL database.

Table 4 shows the categories created and the total training data that was loaded to the database for each of the categories.

Table 4: Classifier Training Data Count

Category	Cluster	Programmes	System	Application	Login	Registration	Payment
Number	41	37	15	17	10	35	68

4.5 Evaluation of the Prototype

To decide determine whether the prototype created was accurately capturing the required categories, it was necessary to evaluate it. The result of the evaluation was important in deciding how reliable the prototype was. Assessment of the prototype involved the following;

1. Comparative analysis of classification of issues
2. Calculation of accuracy, precision, recall and F-Score

4.5.1 Comparative Analysis

This involved the following;

1. Selection a test data set to be used for comparing the categorization of issues by the system to those by the customer support personnel, Table 5 illustrates the data set selected.
2. Cleaning and saving the test data set; this involved ensuring the issue and response text sent where saved in different fields of the check_emails table.
3. Manually categorizing all the responses sent by the user support personnel into the predefined response categories in the response_categories table.
4. Running the application on the available text data and automatically categorizing the emails responses based on the predefined categories.
5. Generating a bar graph to compare the differences in the categorizations.

The prototype validation component was built on Ruby and PHP with MYSQL database. The backend Ruby/MYSQL component is used to classify the responses generated by the system. The front end PHP/MYSQL is a charting script that was used to generate the graphs of the comparison.

4.5.1.1 The Test Data Set

From the sample data a total of 74 data sets were used as the test set from the following categories;

Table 5: Validation Test Data Sets

Category	Cluster	Login	Payment	Programmes	Registration	System	**Total**
Number	10	6	32	14	11	1	**74**

4.5.2 Accuracy, Precision, Recall and F-Score

In order to visualize the performance of the algorithms used confusion matrices were created for each of the algorithms. In the field of machine learning, a confusion matrix, also known as a contingency table or an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class (Wikipedia, 2014). Let us define an experiment from P positive instances and N negative instances for some condition. The four outcomes can be formulated in a 2×2 contingency table or confusion matrix, as shown by Table 6.

Table 6: Confusion Matrix

		Actual Outcome	
	Total Population	**Condition positive**	**Condition negative**
Expected Outcome	**Test outcome positive**	True Positive (**TP**) eqv. with hit	False Positive (**FP**) eqv. with false
	Test outcome Negative	True Negative (**TN**) eqv. with correct rejection	False Negative(**FN**) eqv. with miss

True positives **(TP)** are relevant items that we correctly identified as relevant. True negatives **(TN)** are irrelevant items that we correctly identified as irrelevant. False positives (**FP**) are irrelevant items that we incorrectly identified as relevant. False negatives (**FN**) are relevant items that we incorrectly identified as irrelevant.

Given these outcomes, the following metrics can be defined:

1. **Accuracy**: Measures the percentage of inputs in the test set that the classifier correctly labeled. This is defined as (TP + TP)/N
2. **Precision**: Indicates how many of the items that we identified were relevant; TP /(TP+FP).
3. **Recall**: Indicates how many of the relevant items that we identified; TP/ (TP+FN).
4. **The F-Measure (or F-Score):** Combines the precision and recall to give a single score, is defined to be the harmonic mean of the precision and recall: (2 × Precision × Recall) / (Precision + Recall).

A confusion matrix is a table where each cell [i,j] indicates how often label j was predicted when the correct label was i. Thus, the diagonal entries (i.e., cells |ii|) indicate labels that were correctly predicted, and the off-diagonal entries indicate errors(Bird et al., 2009).

Cross-Validation

To evaluate the models the original corpus was divided into N subsets called folds. The model was then trained with all the data except that in the fold then tested on the fold.

Pseudo Code

The following illustrates the pseudo code for the component used to create the confusion matrix and to cross validate the model;

```
1. Perform k-fold cross-validation and return a confusion matrix involves steps 2 to 10 .
    2. Load corpus # Training data
    3. Assign the number of folds that corpus will be sub-divided into
       or
    4. Assign the percentage of data in each fold that will be used to test the classifier
```

5. Initialize the classifier
6. Assign the key from the corpus that will be used as the category
7. Assign the key from the corpus that will be used as the text
8. Train the classifier
9. Initialize the confusion matrix with the method that determines whether the classification is fp, fn, tp or tn.
10. Classify the sample data assigned for testing

Implementation

The cross-validation component was implemented in Ruby with the cross_validation libraries calling data that had been saved in the database specifically the categories and categories_data tables.

5 RESULTS AND DISCUSSION

This chapter presents and discusses the results obtained from study.

5.1 Presentation of Results

The automatic customer support interpreter component of the conversational agents was designed and developed to use either NB or LSI classifiers. The following section illustrates the typical responses generated by the application to user requests using either of the classifiers, more sample emails and responses can be viewed in Appendix B.

Example 1

User Email

I would like to know what your website refers to as cluster weights . I would also like to know how you arrived at the latter. Thanks in advance

Response by Personnel

the cluster weight has being calculated from your grade and it shows your points for the various cluster groups. it was calculated using the new formula at http://jab.uonbi.ac.ke/node/61

Response by System

Subject: Cluster Points Inquiry

Please send us your VALID KCSE index number so we can verify your registration details. The following is the procedure for viewing your cluster:

i. Visit the page https://kuccpsapp.uonbi.ac.ke/admission_inquiry.php to verify that you qualify for placement. Enter your details in the form provided and submit if you qualify, a list of your previous choices, weighted clusters will appear.

ii. If you would like to know how each cluster is calculated simply click on the decimal value and you will see how it was generated

iii. You should then use the previous years cutoff point to compare with your cluster points for a particular cluster to check whether or not you qualify to apply for the programmed.

iv. To view only those programmes which you qualify to apply based on their minimum subject requirement, cluster requirements and mean grade requirement, ensure you register using the registration form https://kuccpsapp.uonbi.ac.ke/ then proceed to click on View Clusters -> Then click on any of the active cluster links

PLEASE NOTE: v. To view general information on the new formular visit the page https://kuccps.uonbi.ac.ke/node/61

vi. There are no cut off points released for the current admission, however you can use the previous years cutoff points to guide on you if you qualify for admission for more competitive programmes.

Example 2

User Email

Why is it that when i fill in my information for registration they tell me that it is invalid

Response by Personnel

send your details so that we can assist you.Index number KCSE KCPE birtth certificate number and KCPE year.

Response by System

Registration Procedure

Please send us your VALID KCSE index number so we can verify your registration details. Ensure you are using the following details to register Your Birth Certificate or Your Year of birth, Gender, KCPE index number and KCPE year.

If the application is still unable to validate you when you use your KCSE Index number as appearing on you result slip then you have NOT qualified to register and apply for degree programmes

5.2 Conversational Agent Applying Naïve's Bayes Algorithm

To determine which classification to assign to text the classifier assigns weights to all the categories available based on the training data it then proceeds to select the category with the highest weight. Table 7 illustrates the weights assigned for **Example 1**, the selected category is cluster which has the largest value.

Table 7: NB Classification Weights for Example 1

Category	Cluster	Programmes	System	Application	Login	Registration	Payment
Weight	-199.545	-202.957	-207.979	-211.172	-221.111	-241.111	-254.27

5.2.1 *Application Run Statistics*

Training

The classifier using NB took 1.0 seconds to train with 223 data sets. Table 8 shows the application run statistics using the NB algorithm, from these statistics it can be estimated that the application takes 0.4 seconds to handle each request.

Table 8: NB Application Run Statistics

emails	agents	responded	escalated	ignored	pending	Duration (Seconds)	Duration Per Request (Seconds)
242	242	239	0	0	3	108	0.446281
242	242	239	0	0	3	94	0.38843
242	242	239	0	0	0	87	0.359504
821	821	812	0	0	0	324	0.394641

5.2.2 *Validation of the Results*

A comparison of the responses generated by the system versus those sent by the support personnel yielded the graph in Figure 20 which illustrates that out of the 74 test emails that were used to evaluate the system from the different categories (Table 5), only 12 percent were classified in the same category as those generated by support staff.

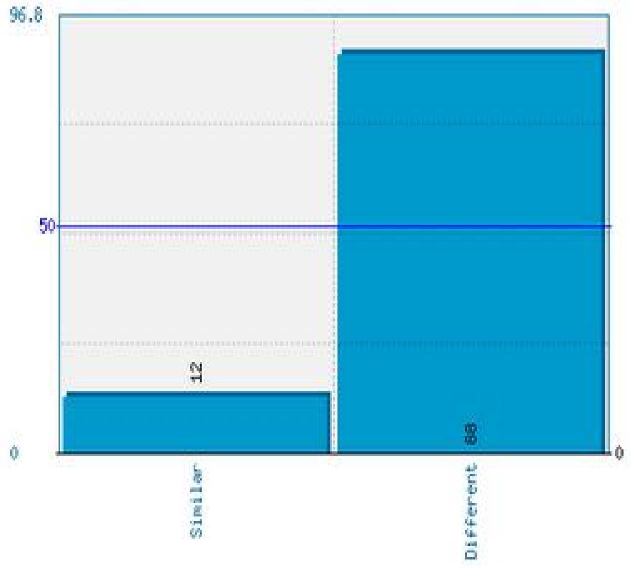


Figure 20: Comparison of System Versus User Responses Using NB

Figure 20 illustrates that with NB the error rate was approximately 88 percent which for a real world application is unacceptable, this necessitated the implementation of another algorithm with improved accuracy.

5.2.3 Accuracy and F-Score

Using various number training data sets cross-validation of the data sets was carried out across 5 folds of data and the mean of the validation calculated as illustrated by Table 9.

Table 9: NB Validation Results

Parameter	**Accuracy**	**Precision**	**Recall**	**F-score**
Results (70 data sets)	0.942	0.942	1.0	0.970
Results (130 data sets)	0.953	0.953	1.0	0.976
Results (200 data sets)	0.925	0.925	1.0	0.961
Mean	0.94	0.94	1	0.969

5.3 Conversational Agent Applying LSI Algorithm

To determine which classification to assign to text the LSI classifier uses a voting system to categorize based on the categories of other text. It takes content and finds other text that is semantically "close", returning an array of documents or text sorted from most to least relevant. Table 10 illustrates the weights for **Example 1**, the selected category in this case is cluster**.**

Table 10: LSI Classification Weights for Example 1

Category	Cluster	Programmes	System	Application	Login	Registration	Payment
Weight	14.4572	5.01403	0.165699	3.43984	0.815106	1.81205	1.62943

5.3.1 Application Run Statistics

Training

The classifier implemented with LSI was found to utilize exhaust system memory (RAM) and CPU in creation of the indexing vector, consequently it was unable to complete the training process with all the available training data. To ensure that the classifier was able to complete training a maximum threshold of 47 data sets per category was employed, it then trained in 21.0 seconds with a total of 200 data sets.

Application Run

Table 11 shows the application run statistics using the LSI algorithm, from these statistics it can be estimated that the application takes 0.4 seconds to handle each request.

Table 11: Application Run Statistics

emails	agents	responded	escalated	ignored	pending	Duration (Seconds)	Duration Per Request (Seconds)
242	242	239	0	0	3	108	0.44628099
821	821	812	0	0	0	324	0.39464068
824	824	824	0	0	0	79	0.09587379

5.3.2 Validation of the Results

A manual classification and comparison of the responses generated by the system versus those sent by the user yielded Figure 21 which shows that out of 74 emails that were tested 74 percent were classified in the same category as those sent by users using LSI algorithm and 26 percent were classified in a different category.

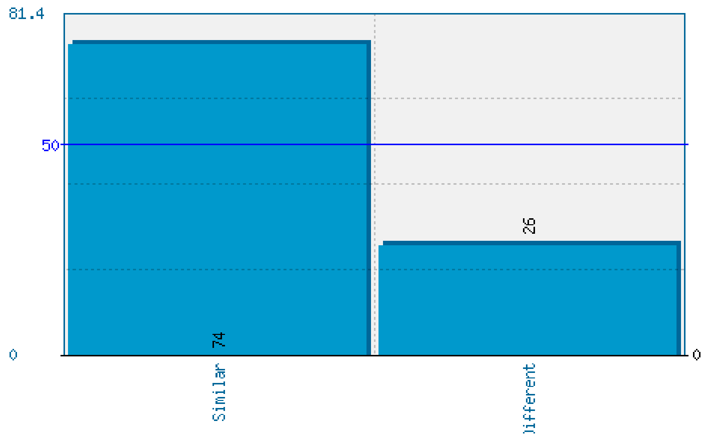


Figure 21: Comparison of System versus User Responses Using LSI

5.3.3 Accuracy and F-Score

Using various training data sets divided into 5 folds the results in Table 12 where acquired.

Table 12: LSI Validation Results

Parameter	**Accuracy**	**Precision**	**Recall**	**F-score**
Results (70 data sets)	0.9	0.9	1.0	0.947
Results (130 data sets)	0.884	0.884	1.0	0.938
Mean	0.892	0.892	1	0.9425

5.4 Discussion of Results

The conversational agents were initially implemented with NB algorithm, Figure 22 shows the error rate which was approximately 88 percent.

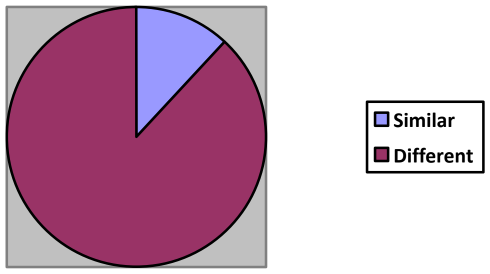


Figure 22: Pie-Chart of NB CA Comparison

The 88 percent error rate was considered unacceptable for a real world application and several techniques were used to try to improve the accuracy of the classifier these included;

1. Stemming this is whereby if you have the terms programmer, programming and program the stemmer reduces this to a single stem example program.
2. Manual training data optimization and cleaning; removal of all words and symbols that would interfere with the classifier accuracy.

The performance of the classifier did not improve necessitating the implementation of another classification algorithm thought to be more accurate. LSI algorithm selected because it has proven to be accurate even when the training data is minimal. With LSI using only 200 training data the algorithm performed better than NB with an error rate of 26 percent and accuracy of 74 percent as illustrated by Figure 23.

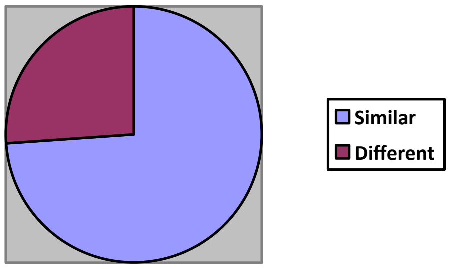


Figure 23: Pie-Chart of LSI CA Comparison

The results (Table 9) demonstrate that the NB classifier has a mean accuracy of 94 percent, precision 94 percent, recall 100 percent and F-score of 97 percent. These results acquired are quite contentious especially given that the algorithm was not able to accurately categorize 88 percent of the test data, this implies that there must be an issue with either the training data set or the algorithm used to calculate the parameters. The results (Table 12) demonstrate that the LSI classifier has a mean accuracy of 89.2 percent, precision 89.2 percent, recall 100 percent and F-score of 94 percent. These results for this algorithm appear to be more realistic especially given the comparison results illustrated in Figure 21.

NB classifier was shown to train in 1 second with 223 training data sets while LSI trained in 21 seconds with 200 datasets as shown by Figure 24. Addition of data sets to the LSI classifier resulted in the classifier utilizing too many system resources resulting in a hanging of the application.

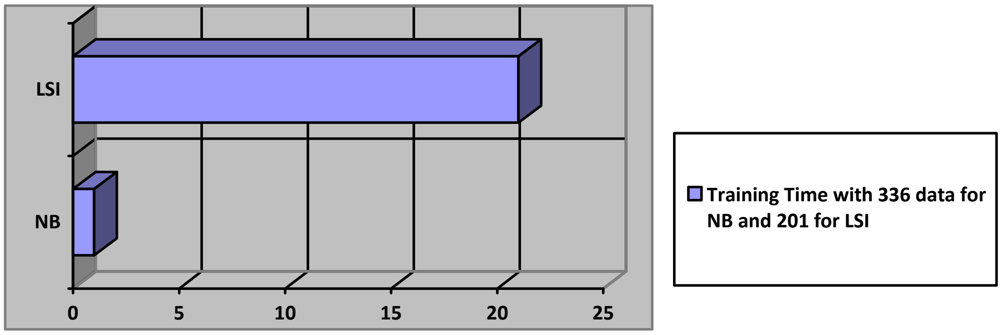


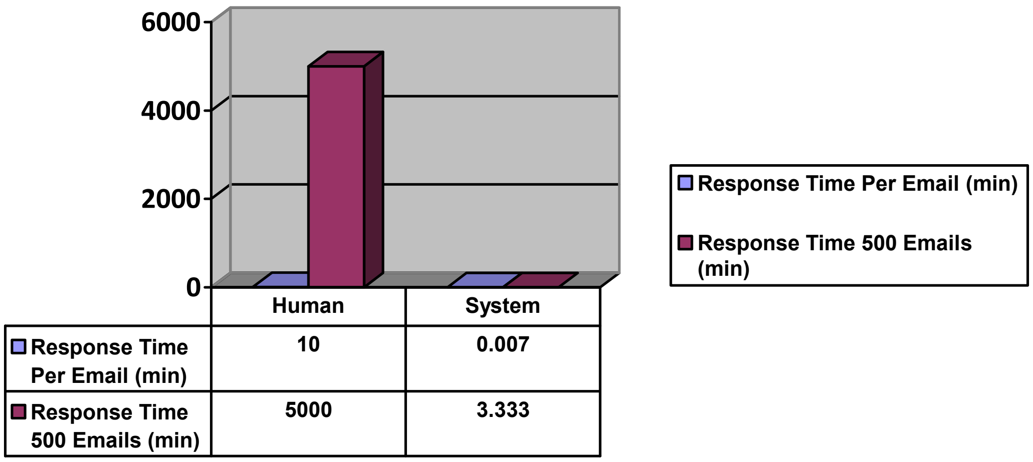
Figure 24: Classifier Training Times

NB classifiers were found to be generally easy to understand and their training extremely fast, requiring only a single pass through the data if all the attributes are discrete. In addition they are robust to irrelevant attributes and classification taking into account evidence from many attributes to make the final prediction. Unfortunately they require strong independence assumptions and when these are violated, the achievable accuracy may deteriorate and will not improve much as the database size increases (Kohavi, 1996). This was illustrated by the poor performance of the classifier with the available training data. The results illustrate that that though LSI classifiers are not as small as NB they may provide more accuracy, flexibility, fast search and clustering detection as well as semantic analysis of the text that theoretically simulates human learning. In this study the LSI classifiers were found to have better accuracy than NB classifier even when utilizing a subset of the training data.

The results obtained in this study illustrate that the classification accuracy depends not only on the classification algorithm employed but also on the quality of training data used. In the event that noisy dataset is used then it is inevitable that the accuracy of the algorithm will be compromised.

The application was found to handle each request in approximately 0.4 seconds regardless of the algorithm used this involved the preprocessing, categorization and response generation, with the polling time to the email account set to 10 seconds or more depending the time set by the user.

Support personnel handle each request in approximately 10 minutes this includes opening the email, reading it, querying the database and generating a suitable response or action then executing or sending the email. If 500 emails were received every day in the university online application Figure 25 illustrates that a support staff would take approximately 5000 minutes or 83.3 hours to handle all these requests. To ensure all the requests were handled effectively in a single day 4 support personnel staff members would be required. However the system takes 3 minutes to handle the 500 emails this demonstrates how effective the application would be in handling routine requests or inquires.



	Human	System
🟦 Response Time Per Email (min)	10	0.007
🟥 Response Time 500 Emails (min)	5000	3.333

Figure 25: Support Staff Versus System Email Response Times

6 CONCLUSIONS AND FUTURE WORK

6.1 Achievements

The following indicates the objectives of this study and how they were met;

1. To identify and analyze techniques used in automating customer support and select suitable technique for creating a customized automatic customer support system. Desk research was done to identify and study existing techniques for automating customer support and the most suitable for the study was selected.
2. To develop a customized automatic customer support system using the selected technique. To meet this objective an application was designed and implemented in RUBY and PHP that incorporated multiple components from conversational agents which use NB or LSI classification algorithms to categorize issues into predefined categories, to email retrieval and sending components and web based interface component. In addition the researcher created and optimized multiple training data sets for categorizing emails into predefined categories. These data sets can be used with any algorithm as they are in a format that is easily interpretable.
3. Conduct a comparative analysis of the performance of the system developed with human customer support. To achieve this objective two system components were created; the first for comparing the responses generated when processing each email and the other for generating the graphs that illustrate the various comparisons. The categorization of responses created by staff was done manually.

The end product of this study was an application that can be easily customized and extended for any industry requiring minimal code modifications.

6.2 Limitations

The study faced several limitations these included;

Data availability; due to the volumes of emails received daily in online support email account the mail administrators have to purge the account from time to time hence drastically reducing the available sample data to be used for training the classifier and validating the application.

Language processing; the application was plagued with issues pertaining to language processing; firstly the inaccuracy of NB which was then substituted with LSI. Unfortunately LSI requires significant computer resources that is RAM greater or equal to 1 GB and needs to be optimized with the GSL library which unfortunately proved problematic to install on windows.

Time; the application required a lot of time to go through all available data to create training data for the two classifiers and to clean the emails sent by personnel in order to allow comparison to those generated by the application.

Power; having relocated to Uganda the researcher found that the region of residence was plagued with electricity outages and blackouts that resulted in delays in achieving set targets.

6.3 Conclusions

During the analysis of the support emails sent and received some issues were noted with the responses generated by user support personnel these included; grammatical errors in emails sent to customers, some queries sent were not understood and the staff simply resent a previous email assuming it would be of assistance to the applicants and the personnel were only able to deal with one issue at a time; basically if a applicants sent an email saying they were unable to register the staff would only resolve that issue and not any other issues example; missing results, payment issues and lost information resulting in the applicants sending multiple emails on different occasions to ensure all their issues are resolved. It was also noted that many of the emails received were quite routine in nature requiring the same response or action by the support staff.

The application developed can be built upon to handle routine user requests pertaining to online information systems that may not require human intervention. The conversational agents can be able to resolve multiple issues noted within short spans of time as note approximately 0.4 seconds per issue. This application can be used in applications that receive high volumes of support requests and have few users to respond.

LSI CA was found to accurately categorize 74 percent of the test data set, this can be greatly improved by cleaning the training data further and introducing more training data after ensuring the server running the application has been optimized to reduce LSI training time. NB classifier accuracy was found to be 94 percent and LSI 89.2 percent respectively from the comparison results there seems to be a discrepancy in the returned accuracy as LSI seems to accurately categorize the test data set.

6.4 Future Work

Future studies should focus on improving the classification algorithm accuracy especially in the case of Naïve's Bayes which is faster than LSI and uses less system resources. The studies could concentrate on either optimization of available training data and further customization of the stop words that are omitted from the processed text. These studies should also study and improve the cross validation library and functions created in order to return realistic values that compare to the comparison results.

Future studies should also investigate and implement handling conversations whereby once a user responds to a system generated email the system simply processes the response sent and not the entire email.

7 APPENDIX A: SAMPLE EMAILS

This section holds samples of the various emails retrieved from the applicants. In order to reduce the content displayed the footer has been omitted from all emails displayed. To protect applicant data, private information has been replaced with dummy data in this section.

Sample 1: Category Payment Details

```
From: "JAB" <jabonline@uonbi.ac.ke>
To: <abc@gmail.com>
Cc: <jabonline@uonbi.ac.ke>
Subject: RE: Money send to wrong account
Date: Fri, 24 May 2013 20:30:28 +0300
MIME-Version: 1.0
Content-Type: text/plain;
        charset="utf-8"
Content-Language: en-us

Your payment was successful, kindly proceed with jab revision
Rgds
Jab

-----Original Message-----
From:abc@gmail.com [mailto:abc@gmail.com]=20
Sent: Tuesday, May 21, 2013 7:27 AM
To: jabonline@uonbi.ac.ke
Subject: Money send to wrong account
Transaction code-1234567,date-20th may,amount paid-800,phone =
no-123456789,index no-25533308053
```

Sample 2: Category Cluster Details

```
From: "JAB" <jabonline@uonbi.ac.ke>
To: <abc@ovi.com>
Cc: <jabonline@uonbi.ac.ke>
Subject: RE: How to calculate weighted cluster point
Date: Fri, 24 May 2013 19:33:20 +0300
MIME-Version: 1.0
Content-Type: text/plain;
        charset="utf-8"
X-Mailer: Microsoft Office Outlook 12.0
Content-Language: en-us

Go to http://jabonline.uonbi.ac.ke/admission_enquiry and fill in the =
details and then you will get your weighted cluster point. Click on the =
value to show you the calculation
Rgds
Jab

-----Original Message-----
From: abc1@ovi.com [mailto:abc1@ovi.com]=20
Sent: Monday, May 20, 2013 11:15 PM
```

```
To: jabonline@uonbi.ac.ke
Subject: How to calculate weighted cluster point

Show me how i can get raw cluster performance index<r>,sum of the =
maximum performance index<m>,aggregate performance index<api> and =
summation performance index<spi>.my grades are,=20
Eng B Kisw D+ Maths A Bio A Chem A Histo A Agric A Geog B+
show me so that i can help many others please.
----------
Sent from my Nokia phone
```

Sample 3: Category Programme Details

```
From: "JAB" <jabonline@uonbi.ac.ke>
To: "ABC" <abc@yahoo.com>
Cc: <jabonline@uonbi.ac.ke>
Subject: RE: Inquiry
Date: Fri, 24 May 2013 19:17:17 +0300
MIME-Version: 1.0
Content-Type: text/plain;
	charset="utf-8"
Content-Transfer-Encoding: quoted-printable
X-Mailer: Microsoft Office Outlook 12.0
Thread-Index: Ac5VmJtYm3FGWW7vRaiWFxSaaj1ZtQDARK5A
Content-Language: en-us

There is no direct way of knowing if you have qualified but you can use =
the previous cut off points to guide you on the degree programmes that =
are more competitive
Rgds
Jab=20
-----Original Message-----
From: ABC [mailto:abc@yahoo.com]=20
Sent: Monday, May 20, 2013 11:29 PM
To: jabonline@uonbi.ac.ke
Subject: Inquiry

Dear Sir/Madam,
Following the review of Degree Choices, I am requesting guidance since the cluster weight is written respectively for each
course, how do I know if am qualified or should I cross-check with Weight cut-off in each university, for example my cousin
scored As in all subjects but Cluster weight is 45. 582 for the first three choices, but on checking course cut-off it is 46 and
above, does it mean he doesn't qualify ?
Thanks in advance.
Regards
Abc
```

Sample 4: Category Registration Details

```
From: "JAB" <jabonline@uonbi.ac.ke>
To: "'ABC Mutai'" <abc@yahoo.com>
Cc: <jabonline@uonbi.ac.ke>
Subject: RE: inquiry,urgent
Date: Fri, 24 May 2013 20:34:10 +0300
MIME-Version: 1.0
Content-Type: multipart/alternative;
        boundary="----=_NextPart_000_0000_01CF9A9C.37BAD300"
X-Mailer: Microsoft Office Outlook 12.0
Thread-Index: Ac5V28h6SZsRUpDYQcma4P+8dAz4fwCyOJHg
Content-Language: en-us

This is a multipart message in MIME format.

------=_NextPart_000_0000_01CF9A9C.37BAD300
Content-Type: text/plain;
        charset="us-ascii"
Content-Transfer-Encoding: 7bit

Kindly register here
http://jabonline.uonbi.ac.ke/applicant_information?applicant_action=applicant_initial_login  and ensure you put the year in full
e.g 2012

Rgds
Jab

 From: ABC Mutai [mailto:abc@yahoo.com]
Sent: Tuesday, May 21, 2013 7:29 AM
To: jabonline@uonbi.ac.ke
Subject: inquiry,urgent

I  have paid the required amount to revise courses but when i try to
register it direct me to provide correct information. please help on how the
year format YYYY is written
```

8 APPENDIX B: SYSTEM RESPONSES GENERATED

This section holds samples of the various emails generated for the different categories of issues. In order to reduce the content displayed the footer has been omitted from all emails displayed. To protect personal data, private information has been replaced with dummy data in this section.

Sample 1: Category Programme Details

```
Delivered-To: abc@gmail.com
Received: by 10.107.16.226 with SMTP id 95csp2796ioq;
        Wed, 5 Nov 2014 23:57:32 -0800 (PST)
From: p58.76753.2012.2@gmail.com
X-Google-Original-From: P58.76753.2012.2@gmail.com
Date: Thu, 06 Nov 2014 10:57:30 +0300
To: abc@gmail.com
Message-ID: <545b29ea97bac_4ef91405c58647db@Asymptotic.mail>
Subject: Programme Details
Mime-Version: 1.0
Content-Type: text/plain;
 charset=UTF-8
Content-Transfer-Encoding: 7bit


    Please send us your VALID KCSE index number so we can verify your registration details.  Please visit the website
http://jab.uonbi.ac.ke/ and use the search option to view details on all available programmes.
        Kindly note that the following:
        i.   Under each university the programmes available for application are listed, if a programme is not listed then it is not
available for application.
        ii.  The cut offs for the programmes are not available only those for 2012 are available and these should be used as a
guide to check the competitive programmes.
             These cutoffs can change for this year depending on average performance and capacity available.
        iii. You should only apply for programmes which you meet the minimum subject requirements, cluster requirements and
which you have a weighted cluster near, equal or greater than the previous year. You should visit the page
https://kuccpsapp.uonbi.ac.ke/admission_inquiry.php to determine the amount, paybill number and account you are required
to pay to.
```

Sample 2: Category Payment Details

```
Delivered-To: abc@gmail.com
From: p58.76753.2012.2@gmail.com
Date: Thu, 06 Nov 2014 10:59:52 +0300
To: abc@gmail.com
Subject: Payment Inquiry
Mime-Version: 1.0
Content-Type: text/plain;
 charset=UTF-8
Content-Transfer-Encoding: 7bit
```

Dear BSCD JOSPHAT ,

The system indicates that you have already registered.

PLEASE NOTE: You should visit the page https://kuccpsapp.uonbi.ac.ke/admission_inquiry.php to determine the amount, paybill number and account you are required to pay to. You have paid LESS than the required amount. Please pay the remaining balance indicated on the inquiry page. You should visit the page https://kuccpsapp.uonbi.ac.ke/admission_inquiry.php to determine the amount, paybill number and account you are required to pay to.

Sample 3: Category System Access

```
Delivered-To: abc@gmail.com
From: p58.76753.2012.2@gmail.com
Date: Thu, 06 Nov 2014 07:39:24 +0300
To: abc@gmail.com
Subject: System Access
Mime-Version: 1.0
Content-Type: text/plain;
 charset=UTF-8
Content-Transfer-Encoding: 7bit
```

Please send us your VALID KCSE index number so we can verify your registration details. Ensure you are using a good internet connection, Mozilla as your default browser and javascript is enabled on your browser. The following is the procedure for application:

i. Visit the page https://kuccpsapp.uonbi.ac.ke/admission_inquiry.php to verify that you qualify for placement. Enter your details in the form provided and submit if you qualify, a list of your previous choices, weighted clusters will appear.

ii. Using results from i. determine if you would like to revise your choices, if so pay the indicated amount to the indicated Mpesa Pay bill number and Account

iii. Visit the website https://kuccps.uonbi.ac.ke/ to view and selected the available programmes that you may qualify for, note down programme codes for 3 similar (optional) and three others you may be interested in applying for.

iv. Once you have received a payment confirmation text visit https://kuccpsapp.uonbi.ac.ke and register by providing the requested information and entering a password.

v. You can then proceed to enter your selected programmes in the Application form provided, remember to save.

vi. Review the choices saved and log out.

REMEMBER YOU CAN CHANGE THE CHOICES YOU MAKE AS MANY TIMES AS YOU WISH DURING THE REVISION PERIOD

9 APPENDIX C: CLASSIFIERS

This section data used in specified ways and arrays or vectors that were generated by the two algorithms

9.1 STOP WORDS

This section contains some of the words which the classifiers are required to omit from text before they can classify

2gt a able about ascertain admissions after again all along appreciate are also an and as assist assistance at back be because become became been beg board but by came come can cannot cant choice choose chose chosen choosen click consider considered considers couldnt dear decision decide detail did didn didnt display displays displaying do doesnt does dont done each easy easier easily enter even eventually ever every fill filling filled find finding first follow followed follower for from gave get give given go got had hadnt has hasnt have havent havenot hello help her hereby herby here him hi how i if in inform infor information intro introduction into instructions is isnt it itll jab join joint just kindly know knew last least left leaving like link madam made met meet most my myself mine need new no not notify notifies notifys now of on or other otherwise please procedure problem problems rather re re: read receive received replied reply run ryou said save see send self should sinc since sir so soo some sometime sometimes suppose supposed supposes still take takes taken took time till th than thank this that the their there they them then those through though to told too true tried try unable until update url us via view wait web website were what when whether whenever which while why will with within without write wrote yes yet you ypu your youll rgdsjab http httpjabonline jab jabonline here regard regards enquiry inquiry uonbi keadmission_enquiry gmail com mail monday tuesday wednesday thursday friday saturday sunday message follow admission_enquiry original mailto sent amto mail ke kesubject revisingthank revis courseshi nokia phone revision mail mai admission check course courses revisin revising revision following yesterday tommorrow tomorrow yahoo revisionrgdsjab was account ableto amessage office proceed proceedwith be provide

9.2 NB CLASSIFIER

This section illustrates the hash generated by the NB classifier during each run of the application that is then used in classification of text.

```
#<ClassifierReborn::Bayes:0x00000004b84d50 @categories={:Systemaccess=>{:slow=>2, :speed=>1, :access=>5,
:system=>4, :hang=>1, :load=>5, :page=>5, :login=>1, :choic=>1, :error=>1, :try=>2, :veri=>1, :modul=>1, :miss=>1,
:captcha=>1, :open=>2, :delai=>1, :activ=>1, :center=>1, :number=>1, :submit=>2, :appllic=>1, :pop=>1, :menu=>1, :prompt=>1,
:show=>1, :webpag=>1, :browser=>1, :need=>1, :resend=>1, :site=>1, :offlin=>1},
:Login=>{:login=>3, :system=>2, :authent=>2, :wrong=>1, :index=>2, :registr=>1, :access=>1, :deni=>1, :be=>1, :provid=>2,
:invalid=>1, :password=>2, :try=>1, :regist=>1, :candid=>1, :point=>4, :paid=>1, :respons=>1, :get=>1, :number=>1,
:minimum=>1, :cut=>1, :off=>1, :femal=>1, :forgotten=>1, :log=>1, :manag=>1, :wai=>1},
:Paymentdetails=>{:payment=>31, :go=>2, :success=>2, :system=>2, :monei=>9, :refund=>3, :mobil=>1, :wrong=>20,
:blunder=>1, :pai=>7, :fee=>10, :onli=>1, :enter=>1, :nine=>2, :digit=>3, :instead=>7, :eleven=>2, :transact=>16, :code=>11,
:amount=>15, :paid=>26, :be=>2, :number=>27, :us=>15, :mpesa=>9, :overpay=>1, :excess=>1, :dure=>1, :error=>1,
:instruct=>1, :current=>1, :index=>24, :consider=>1, :incorrect=>2, :indexno=>3, :correct=>4, :balanc=>3, :todai=>1, :invalid=>3,
:qualifi=>1, :sai=>2, :quot=>1, :make=>4, :cohort=>2, :student=>2, :who=>2, :minimum=>1, :cut=>2, :off=>2, :point=>2, :,
………..},
:Registrationdetailsinquiry=>{:regist=>17, :registr=>9, :index=>6, :number=>7, :birth=>4, :certif=>3, :be=>3, :provid=>5,
:invalid=>6, :correct=>3, :center=>1, :applic=>1, :onlin=>5, :try=>6, :success=>2, :procce=>2, :page=>2, :reset=>2, :right=>1,
:access=>1, :befor=>2, :closur=>1, :exercis=>1, :detail=>2, :name=>1, :kcse=>2, :kcpe=>3, :year=>3, :log=>2, :complet=>1,
:password=>6, :us=>2, :must=>2, :login=>1, :claim=>1, :process=>2, :ascertain=>1, :address=>1, :verifi=>1, :cutoff=>1,
:point=>5, :similar=>1, :degre=>2, :section=>1, :qualifi=>1, :revis=>4, :univers=>2, :fail=>1, :attempt=>3,
…………………………………………………….},
:Programmesdetails=>{:cutoff=>13, :point=>45, :cut=>15, :off=>13, :variou=>6, :degre=>12, :enabl=>2, :revis=>6, :cluster=>19,
:offer=>2, :public=>1, :univers=>6, :academ=>1, :year=>11, :shall=>1, :chang=>3, :compar=>1, :form=>1, :where=>3,
:download=>1, :differ=>2, :weight=>6, :onli=>2, :displai=>2, :column=>1, :guidelin=>1, :get=>1, :us=>5, :intak=>1, :previou=>3,
:mean=>2, :qualifi=>3, :engin=>2, :ask=>2, :formula=>4, :admit=>1, :student=>1, :go=>1, :reduc=>2, :program=>2, :consid=>1,
:on=>4, :make=>1, :inform=>1, :decis=>1, :select=>1, :base=>2, :old=>1, :formular=>1, :grade=>1, :would=>5, :pursu=>2,
:medicin=>1, :possibl=>1, :concern=>1, :access=>1, :calcul=>5, ……………},
:Clusterinquiry=>{:cluster=>49, :point=>27, :formula=>6, :weight=>15, :need=>1, :programm=>1, :order=>1, :qualifi=>5, :ani=>1,
:perform=>5, :index=>6, :per=>1, :subject=>8, :calcul=>13, :check=>2, :attain=>1, :entri=>1, :match=>1, :respect=>3, :appli=>3,
:revis=>1, :show=>1, :raw=>1, :sum=>1, :maximum=>1, :aggreg=>2, :grade=>2, :eng=>1, :kisw=>1, :math=>2, :bio=>1,
:chem=>1, :histo=>1, :agric=>1, :geog=>1, :where=>2, :year=>4………..
:Applicationprocess=>{:far=>1, :end=>1, :weight=>2, :point=>4, :differ=>1, :on=>1, :us=>1, :li=>1, :question=>1, :these=>1,
:year=>2, :attain=>1, :qualifi=>1, :applic=>4, :work=>1, :want=>1, :univers=>3, :intak=>1, :tell=>1, :admit=>1, :degre=>6,
:choic=>5, :hard=>1, :select=>2, :cluster=>3, :refer=>1, :thank=>1, :advancegt=>1, :must=>1, :simmilar=>1, :slot=>1, :impot=>1,
:import=>1, :go=>1, :place=>1, :next=>1, :formeri=>1, :requir=>2, :consind=>1, :final=>1, :k
cse=>1, :candid=>1, :who=>1, :revis=>2, :todai=>1, :paid=>1, :fee=>1, :chang=>1, :effect=>1, :previou=>1, :under=>1,
:name=>1, :someon=>1, :assur=>1, :succes=>1, :maximum=>1, :number=>1, :time=>1, :dure=>1, :revisiongt=>1}},
@total_words=1367,
@category_counts={:Systemaccess=>15, :Login=>10, :Paymentdetails=>47, :Registrationdetailsinquiry=>34,
:Programmesdetails=>37, :Clusterinquiry=>40, :Applicationprocess=>17}>
```

9.3 LSI CLASSIFIER

The following illustrates a snippet of the LSI vector generated by the application. The generated vector is quite large and cannot be displayed here. The key for the vector is the categorization text.

```
#<ClassifierReborn::LSI:0x000000039c6528 @auto_rebuild=true, @items={"Enquiry About
Courses"=>#<ClassifierReborn::ContentNode:0x000000039c1730 @categories=[:Applicationprocess], @word_hash={},
@raw_norm=GSL::Vector[ -nan -nan -nan -nan -nan -nan -nan ... ], @raw_vector=GSL::Vector
[ 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 ... ], @lsi_vector=GSL::Vector
[ 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 ... ], @lsi_norm=GSL::Vector
[ -nan -nan -nan -nan -nan -nan -nan ... ]>, "At the far end there are weighted points but different from the ones I used for
2009 here lies my question.Are these the points for this year and should I have attained If they are am I qualified for those
courses"=>#<ClassifierReborn::ContentNode:0x000000039bc780 @categories=[:Applicationprocess], @word_hash={:far=>1,
:end=>1, :weight=>1, :point=>2, :differ=>1, :on=>1, :us=>1, :"2009"=>1, :li=>1, :questionar=>1, :these=>1, :year=>1, :attain=>1,
:qualifi=>1}, @raw_norm=GSL::Vector
[ 2.539e-01 2.539e-01 2.539e-01 4.024e-01 2.539e-01 2.539e-01 2.539e-01 ... ], @raw_vector=GSL::Vector[ 2.650e-01 2.650e-
01 2.650e-01 4.200e-01 2.650e-01 2.650e-01 2.650e-01 ... ], @lsi_vector=GSL::Vector[ 2.615e-01 2.615e-01 2.676e-01 4.203e-
01 2.654e-01 2.666e-01 2.678e-01 ... ], @lsi_norm=GSL::Vector[ 2.507e-01 2.507e-01 2.565e-01 4.029e-01 2.544e-01 2.555e-
01 2.567e-01 ... ]>, "my application is not working"=>#<ClassifierReborn::ContentNode:0x000000039bbf88 @categories =
[:Applicationprocess], @word_hash ={:applic=>1, :work=>1}, @raw_norm=GSL::Vector[ 0.000e+00 0.000e+00 0.000e+00
0.000e+00 0.000e+00 0.000e+00 0.000e+00 ... ], @raw_vector=GSL::Vector[ 0.000e+00 0.000e+00 0.000e+00 0.000e+00
0.000e+00 0.000e+00 0.000e+00 ... ], @lsi_vector=GSL::Vector [ 8.462e-05 8.462e-05 -1.406e-04 -3.860e-05 1.815e-04 -
1.390e-03 -1.046e-03 ... ], @lsi_norm=GSL::Vector[ 5.985e-05 5.985e-05 -9.944e-05 -2.730e-05 1.284e-04 -9.830e-04 -7.399e-
04 ... ]>, ........................................ "cluster revision clusters"=> #<ClassifierReborn::ContentNode:0x00000003946da0
@categories=[:Clusterinquiry], @word_hash={:cluster=>2}, @raw_norm=GSL::Vector [ 0.000e+00 0.000e+00 0.000e+00
0.000e+00 0.000e+00 0.000e+00 0.000e+00 ... ], @raw_vector=GSL::Vector [ 0.000e+00 0.000e+00 0.000e+00 0.000e+00
0.000e+00 0.000e+00 0.000e+00 ... ], @lsi_vector=GSL::Vector[ 1.018e-04 1.018e-04 -4.688e-03 3.919e-03 -1.031e-03 1.558e-
03 2.852e-03 ... ], @lsi_norm=GSL::Vector [ 5.091e-05 5.091e-05 -2.345e-03 1.960e-03 -5.157e-04 7.792e-04 1.427e-03 ... ]>,
"NEW CLUSTER POINT FORMULA"=>#<ClassifierReborn::ContentNode:0x0000000393d458 @categories=[:Clusterinquiry],
@word_hash={:cluster=>1, :point=>1, :formula=>1}, @raw_norm=GSL::Vector [ 0.000e+00 0.000e+00 0.000e+00 5.774e-01
0.000e+00 0.000e+00 0.000e+00 ... ], @raw_vector=GSL::Vector [ 0.000e+00 0.000e+00 0.000e+00 6.309e-01 0.000e+00
0.000e+00 0.000e+00 ... ], @lsi_vector=GSL::Vector [ -3.126e-03 -3.126e-03 -2.562e-03 6.400e-01 -1.959e-02 1.792e-02
4.617e-04 ... ], @lsi_norm=GSL::Vector[ -2.897e-03 -2.897e-03 -2.374e-03 5.931e-01 -1.815e-02 1.661e-02 4.278e-04 ...
]>,........................................................................ @word_list=#<ClassifierReborn::WordList:0x00000003af0ed0
@location_table={:far=>0, :end=>1, :weight=>2, :point=>3, :differ=>4, :on=>5, :us=>6, :"2009"=>7, :li=>8, :questionar=>9,
:these=>10, :year=>11, :attain=>12, :qualifi=>13, :applic=>14, :work=>15, :want=>16, :univers=>17, :intak=>18, :tell=>19,
:admit=>20, :degre=>21, :choic=>22, :hard=>23, :select=>24, :"345"=>25, :cluster=>26, :refer=>27, :thank=>28,
:advancegt=>29, :must=>30, :simmilar=>31, :slot=>32, :impot=>33, :import=>34, :go=>35, :place=>36, :next=>37, :formeri=>38,
:requir=>39, :consind=>40, :final=>41, :"2012"=>42, :kcse=>43, :candid=>44, :who=>45, :revis=>46, :todai=>47, :paid=>48,
:feebut=>49, :chang=>50, :effect=>51, :previou=>52, :under=>53, :name=>54, :someon=>55, :assur=>56, :succes=>57,
:maximum=>58, :number=>59, :time=>60, :dure=>61, :revisiongt=>62, :formula=>63, :need=>64, :programm=>65, :order=>66,
:ani=>67, :perform=>68, :index=>69, :per=>70, :subject=>71, :calcul=>72, :check=>73, :entri=>74, :match=>75, :respect=>76,
:appli=>77, :show=>78, :raw=>79, :sum=>80, :aggreg=>81, :indexmi=>82, :grade=>83, :eng=>84, :kisw=>85, :math=>86,
:bio=>87, :chem=>88, :histo=>89, :agric=>90, :geog=>91, :where=>92, :formswher=>93, :download=>94, :enquri=>95,
:group=>96, :wish=>97, :enquir=>98, :reffer=>99, :entail=>100, :have=>101, :difficulti=>102, :understand=>103, :simplifi=>104,
:abov=>105,...........................................
```

10 APPENDIX D: USER GUIDE

In order to run the application a user would require to follow the following steps to install the application on a machine

1. **System Requirements**

i. Windows or Linux Operating system

ii. 64 bit system

iii. RAM 1GB

iv. CPU 2.4 GHz

2. **Installation Requirements**

i. Apache version >= 2.4

ii. MySQL version >= 5.6

iii. PHP version >= 5.5

iv. PHP extensions;
Windows specific: win32service

v. Ruby version >= 2.0

vi. Ruby gems/libraries; mailman, mysql, sanitize, classifier-reborn, agentdispatcher,gsl, cross_validation
Windows specific; win32
Linux specific; rvm, basic_daemon

3. **Installation**

i. Install Apache, MySQL and PHP, then install and load the indicated PHP extensions on the machine.

ii. Install Ruby on the machine then install all the indicated Ruby gems, on Windows configure the Apache server to run Ruby files. On Linux if using RVM ensure you run the following command before installing any gem;
rvm ruby-(RUBYVERSION)@global
replace the text RUBYVERSION with your preferred Ruby version.

iii. Copy and paste the following functions into the indicated files, if the function already exists replace it with the code;

ruby\lib\ruby\gems\2.0.0\gems\classifier-reborn-2.0.1\lib\classifier-reborn\bayes.rb

```
def classify(text)
      #Code that returns the classification array
     samecount =0
      classificationArray = (classifications(text).sort_by { |a| -a[1] })
      proposedcatweight = classificationArray[0][1]
      classificationArray.each {|key, value|
        #puts "#{key} is #{value}"
        if (proposedcatweight==value)
          samecount +=1
        end
      }
      if (samecount>=2)
        return nocat = ""
      else
       return classificationArray[0][0]
      end
    end
```

ruby\lib\ruby\gems\2.0.0\gems\classifier-reborn-2.0.1\lib\classifier-reborn\lsi.rb

```
def train(category, text)
      add_item text, eval(":#{category}")
    end

def classifycustom( doc, cutoff=0.30, &block )
      icutoff = (@items.size * cutoff).round
      carry = proximity_array_for_content( doc, &block )
      carry = carry[0..icutoff-1]
      votes = {}
      carry.each do |pair|
        categories = @items[pair[0]].categories
        categories.each do |category|
          votes[category] ||= 0.0
          votes[category] += pair[1]
        end
      end
      return votes
    end
```

iv. Extract and copy the application files to the web root of the Apache server.

v. Create 3 databases on the mysql server and import the corresponding MySQL scripts from the autosupport/sql directory as indicated;

   a. online_application: online_application.sql
   b. customer_suppost: customer_support.sql
   c. web: web.sql

vi. Customize the application settings i.e. database and email credentials in the file autosupport/supportapp/settings.rb then save. Ensure the following configurations are set in the setting.rb file;

```
ENVIROMENT ='LIVE'
SHOWEMAILTEXT =false
FUNCTION ='classify'
CLASSIFIER ='NB'
MAXLSI ='47'
CHECKUNIQUE =true
RUNSERVICE =true
CLEARTESTDATA =false
CROSSVALIDATE =false\
```

If you have installed gsl successfully you can change the classifier to LSI

vii. Customize the database, host and file path settings in the file autosupport/admin/settings.rb and Drupal front end settings in the file autosupport/sites/default/settings.rb then save the changes made.

viii. To update or add data sets go to the following web pages;

a. Applicant data: http://localhost/autosupport/admin/jab/admissionslist.php
b. Support data : http://localhost/autosupport/admin/supportupdate/login.php

Details of the data expected in the tables can be found in the design document. You can then proceed to use the available forms to update data.

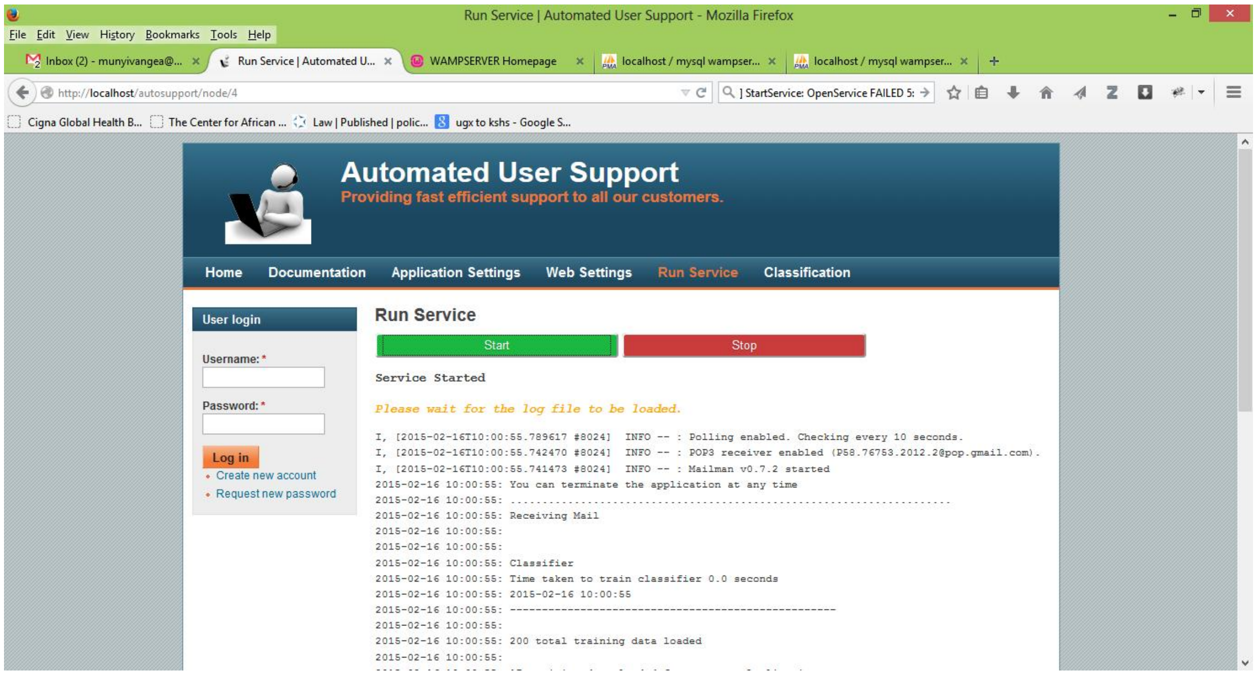
ix. Windows: You will need to register and start the service; open command prompt as administrator and run the following commands;

```
ruby registerservice.rb
sc start supportservice
```

To stop and unregister the Windows the service you will need to run;

```
sc stop supportservice
ruby unregisterservice.rb
```

x. To run the application enter the url http://localhost/autosupport/node/4 on your browser the following should be displayed, click the Start button.



Run Service

```
Service Started

Please wait for the log file to be loaded.

I, [2015-02-16T10:00:55.789617 #8024]  INFO -- : Polling enabled. Checking every 10 seconds.
I, [2015-02-16T10:00:55.742470 #8024]  INFO -- : POP3 receiver enabled (P58.76753.2012.2@pop.gmail.com).
I, [2015-02-16T10:00:55.741473 #8024]  INFO -- : Mailman v0.7.2 started
2015-02-16 10:00:55: You can terminate the application at any time
2015-02-16 10:00:55: ....................................................................
2015-02-16 10:00:55: Receiving Mail
2015-02-16 10:00:55:
2015-02-16 10:00:55:
2015-02-16 10:00:55: Classifier
2015-02-16 10:00:55: Time taken to train classifier 0.0 seconds
2015-02-16 10:00:55: 2015-02-16 10:00:55
2015-02-16 10:00:55: --------------------------------------------------
2015-02-16 10:00:55:
2015-02-16 10:00:55: 200 total training data loaded
2015-02-16 10:00:55:
```

xi. All new emails in the indicated email account or mail directory will be processed and appropriate responses generated and sent to the indicated sender email address.

xii. To view the application run statistics you can visit the page http://localhost/autosupport/admin/supportupdate/application_runlist.php

4. Testing/ Comparison

i. Update the data sets in the table check_emails to use emails for testing purposes ensure the field check_status is set to 0

ii. Load all your test emails in the mail directory that is autosupport/supportapp/mail/new

iii. Customize the application settings in the file autosupport/supportapp/settings.rb, ensure the following configurations are set;

```
ENVIROMENT ='TEST'
FUNCTION ='classify'
CHECKUNIQUE =true
RUNSERVICE =false
CLEARTESTDATA =false
```

CROSSVALIDATE =false

iv. Open command prompt or console as administrator and ensure you are in the directory with the application files that is autosupport/supportapp/ then run the command;

Windows: ruby mailfetchservice.rb

Linux: ruby daemon.rb

v. Customize the application settings in the file autosupport/supportapp/settings.rb, ensure the following configurations are set;

ENVIROMENT ='TEST'

FUNCTION ='classify_responses'

RUNSERVICE =false

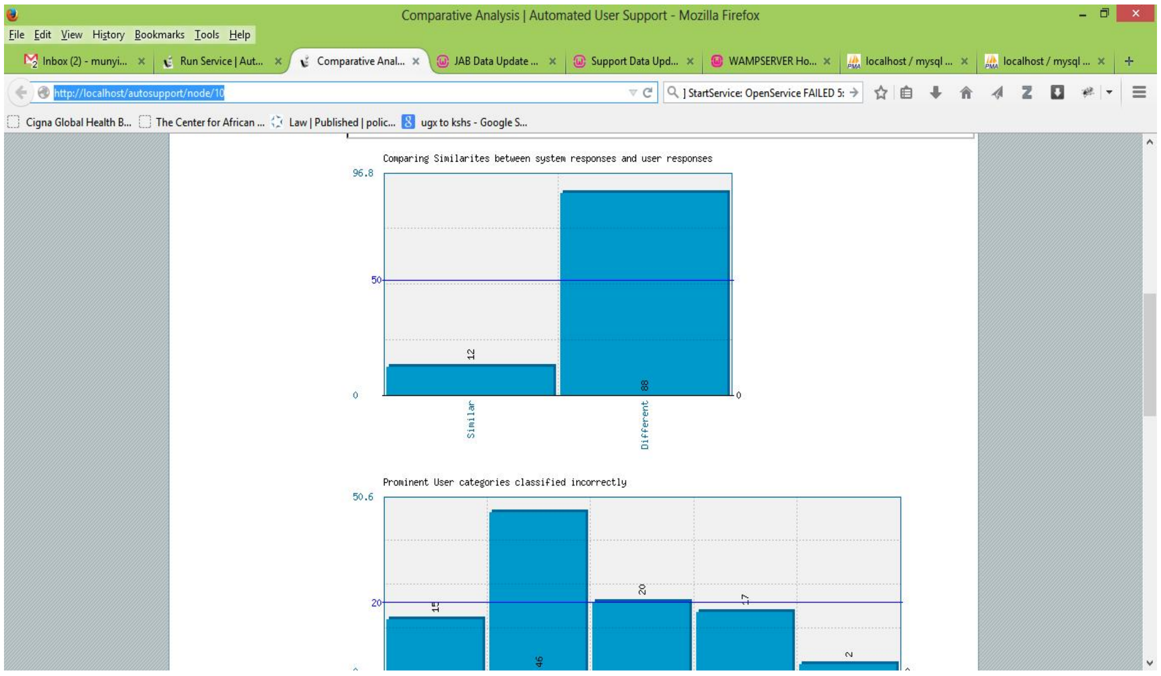
CLEARTESTDATA =false

CROSSVALIDATE =false

vi. Open command prompt or console as administrator and ensure you are in the directory with the application files that is autosupport/supportapp/ then run the command;

ruby testruns.rb

vii. To view generated reports on comparisons visit the page http://localhost/autosupport/node/10 you should then view the following generated graphs;


Comparative Analysis | Automated User Support - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Inbox (2) - munyi...
Run Service | Aut...
Comparative Anal...
JAB Data Update ...
Support Data Upd...
WAMPSERVER Ho...
localhost / mysql ...
localhost / mysql ...
http://localhost/autosupport/node/10
] StartService: OpenService FAILED 5:
Cigna Global Health B...
The Center for African ...
Law | Published | polic...
ugx to kshs - Google S...
Comparing Similarites between system responses and user responses
96.8
50
0
12
88
Similar
Different
Prominent User categories classified incorrectly
50.6
20
15
46
20
17
2

5. Generating Confusion Matrix

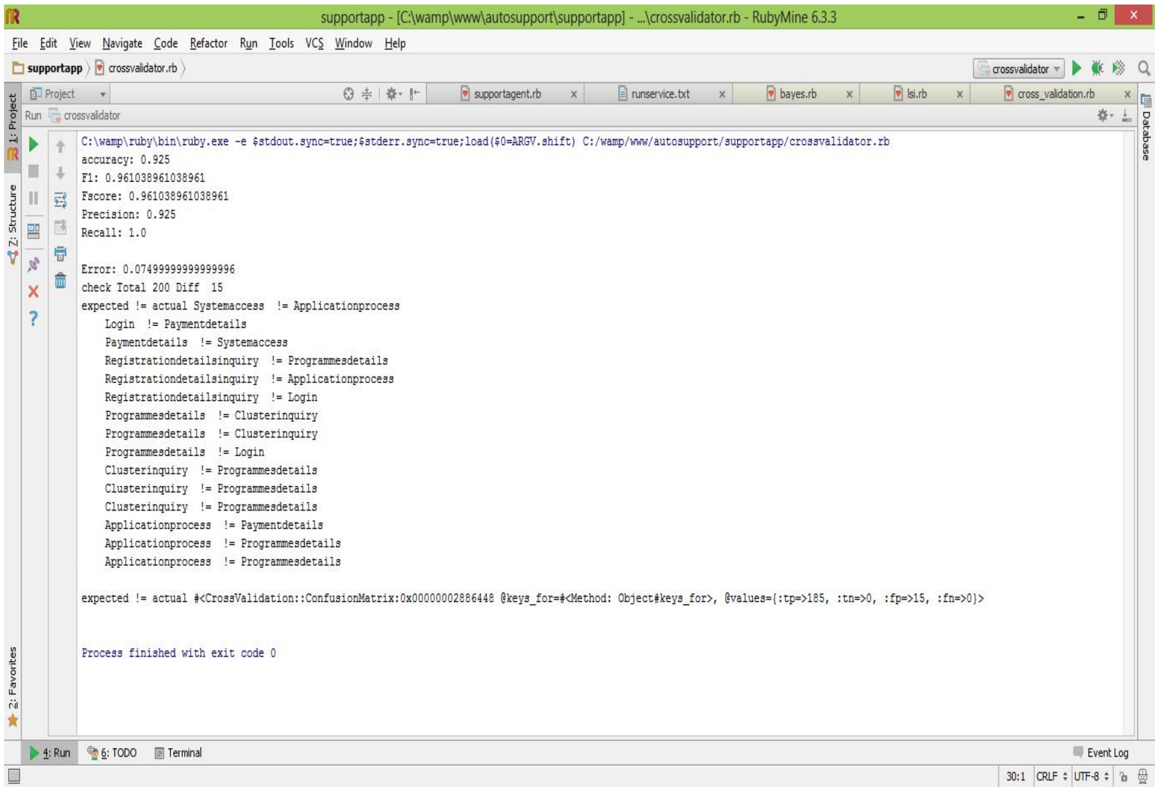
i. Customize the application settings in the file autosupport/supportapp/settings.rb, ensure the following configurations are set;

```
ENVIROMENT ='TEST'
FUNCTION ='classify'
CHECKUNIQUE =true
RUNSERVICE =false
CLEARTESTDATA =false
CROSSVALIDATE =true
```

ii. Open command prompt or console as administrator and ensure you are in the directory with the application files that is autosupport/supportapp/ then run the command;

```
ruby crossvalidator.rb
```

iii. An example of the output of running the script is illustrated below;


supportapp - [C:\wamp\www\autosupport\supportapp] - ...\crossvalidator.rb - RubyMine 6.3.3
File Edit View Navigate Code Refactor Run Tools VCS Window Help
supportapp
crossvalidator.rb
supportagent.rb
runservice.txt
bayes.rb
lsi.rb
cross_validation.rb
Run
crossvalidator
4: Run
6: TODO
Terminal
Event Log
30:1 CRLF UTF-8

```
C:\wamp\ruby\bin\ruby.exe -e $stdout.sync=true;$stderr.sync=true;load($0=ARGV.shift) C:/wamp/www/autosupport/supportapp/crossvalidator.rb
accuracy: 0.925
F1: 0.961038961038961
Fscore: 0.961038961038961
Precision: 0.925
Recall: 1.0

Error: 0.07499999999999996
check Total 200 Diff  15
expected != actual Systemaccess  != Applicationprocess
    Login  != Paymentdetails
    Paymentdetails  != Systemaccess
    Registrationdetailsinquiry  != Programmesdetails
    Registrationdetailsinquiry  != Applicationprocess
    Registrationdetailsinquiry  != Login
    Programmesdetails  != Clusterinquiry
    Programmesdetails  != Clusterinquiry
    Programmesdetails  != Login
    Clusterinquiry  != Programmesdetails
    Clusterinquiry  != Programmesdetails
    Clusterinquiry  != Programmesdetails
    Applicationprocess  != Paymentdetails
    Applicationprocess  != Programmesdetails
    Applicationprocess  != Programmesdetails

expected != actual #<CrossValidation::ConfusionMatrix:0x00000002886448 @keys_for=#<Method: Object#keys_for>, @values={:tp=>185, :tn=>0, :fp=>15, :fn=>0}>


Process finished with exit code 0
```

11 REFERENCES

Aamodt, A., Plaza, E., 1994. Case-based reasoning; Foundational issues, methodological variations, and system approaches. AI COMMUNICATIONS 7, 39–59.

Algorithms, H.C., 2013. A Tutorial on Clustering Algorithms. Online][Cited: 24 04 2008.] http://home. dei. polimi. it/matteucc/Clustering/tutorial_html/hierarchical. html.

Bird, S., Klein, E., Loper, E., 2009. Natural Language Processing with Python. O'Reilly Media, Inc.

BusinessDictionary.com, 2014. What is customer support? definition and meaning [WWW Document]. BusinessDictionary.com. URL http://www.businessdictionary.com/definition/customer-support.html (accessed 6.20.14).

CCK, 2008. Communications Statistics Report 2008 [WWW Document]. Communications Commission of Kenya. URL http://www.ca.go.ke/images/downloads/STATISTICS/Communications%20Statistics%20Report%202008.pdf (accessed 6.20.14).

CCK, 2014. Sector Statistics Report Q2 2013/14 [WWW Document]. Communications Commission of Kenya. URL http://www.cck.go.ke/resc/downloads/Sector_Statistics_Report_Q2_201314.pdf (accessed 6.20.14).

Chang, K.H., Raman, P., Carlisle, W.H., Cross, J.H., 1996. A self-improving helpdesk service system using case-based reasoning techniques. Computers in Industry 30, 113–125.

Chow, D., 2009. Evolution of Information Systems.

Crockett, K., James, O., Bandar, Z., 2011. Goal orientated conversational agents: Applications to benefit society, in: Agent and Multi-Agent Systems: Technologies and Applications. Springer, pp. 16–25.

Customer support, 2014. . Wikipedia, the free encyclopedia.

Dialog system, 2014. . Wikipedia, the free encyclopedia.

Feng, J., Bangalore, S., Rahim, M., 2003. Webtalk: Mining websites for automatically building dialog systems, in: Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on. IEEE, pp. 168–173.

Ghosh, S., Roy, S., Bandyopadhyay, S.K., 2012. A tutorial review on Text Mining Algorithms. International Journal of Advanced Research in Computer and Communication Engineering 1.

GoK, 2007. Vision 2030.

Gray, A., 2011. Web Search and Text Mining.

Ikonomakis, M., Kotsiantis, S., Tampakas, V., 2005. Text classification using machine learning techniques. WSEAS Transactions on Computers 4, 966–974.

Kamruzzaman, S.M., Haider, F., Hasan, A.R., 2010. Text classification using data mining. arXiv preprint arXiv:1009.4987.

Kohavi, R., 1996. Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid., in: KDD. pp. 202–207.

Kruchten, P., 2001. Agility with the RUP. Cutter IT Journal 14, 27–33.

Lama, P., 2013. Clustering system based on text mining using the K-means algorithm: news headlines clustering.

Latent semantic indexing, 2014. . Wikipedia, the free encyclopedia.

Lester, J., Branting, K., Mott, B., 2004. Conversational agents. The Practical Handbook of Internet Computing.

Liddy, E.D., 2000. Text Mining. Bul. Am. Soc. Info. Sci. Tech. 27, 13–14. doi:10.1002/bult.184

Mokaya, J., 2012. Fibre-optic cable Internet helps Kenyan economy [WWW Document]. Sabahionline.com. URL http://sabahionline.com/en_GB/articles/hoa/articles/features/2012/02/20/feature-01 (accessed 6.23.14).

Open Access BPO, 2013. The different types of customer support services - Open Access BPO.

Patra, A., Singh, D., 2013. A Survey Report on Text Classification with Different Term Weighing Methods and Comparison between Classification Algorithms. International Journal of Computer Applications 75, 14–18.

Ramesh, V., Ramar, K., 2011. Classification of Agricultural Land Soils: A Data Mining Approach. Agricultural Journal 6, 82–86. doi:10.3923/aj.2011.82.86

Sawy, O.A.E., Bowles, G., 1997. Redesigning the Customer Support Process for the Electronic Economy: Insights from Storage Dimensions. MIS Quarterly 21, 457–483. doi:10.2307/249723

Segall, R.S., Zhang, Q., Cao, M., 2009. Web-Based Text Mining of Hotel Customer Comments Using SAS® Text Miner and Megaputer Polyanalyst®. SWDSI 2009 141–152.

Song, W., Park, S.C., 2009. Genetic algorithm for text clustering based on latent semantic indexing. Computers & Mathematics with Applications, Proceedings of the International Conference Bio-Inspired Computing-Theories and Applications BIC-TA 2007 Zhengzhou, China 57, 1901–1907. doi:10.1016/j.camwa.2008.10.010

Wang, D., Li, T., Zhu, S., Gong, Y., 2011. iHelp: An Intelligent Online Helpdesk System. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 41, 173–182. doi:10.1109/TSMCB.2010.2049352

Wikipedia, 2014. Confusion matrix. Wikipedia, the free encyclopedia.

Wood, M.F., DeLoach, S.A., 2001. An overview of the multiagent systems engineering methodology, in: Agent-Oriented Software Engineering. Springer, pp. 207–221.

Zukas, A., Price, R.J., 2003. Document categorization using latent semantic indexing, in: Proceedings, Symposium on Document Image Understanding Technology. pp. 87–91.