

AUTOMATED MESSAGE TRIAGE



AUTOMATED MESSAGE TRIAGE:
A PROPOSAL
FOR
SUPERVISED SEMANTIC
CLASSIFICATION OF MESSAGES

By
AMIR TAVASOLI, B.Sc

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Master of Science

McMaster University
© Copyright by Amir Tavasoli, September 2010



MASTER OF SCIENCE (2010)
(Computer Science)

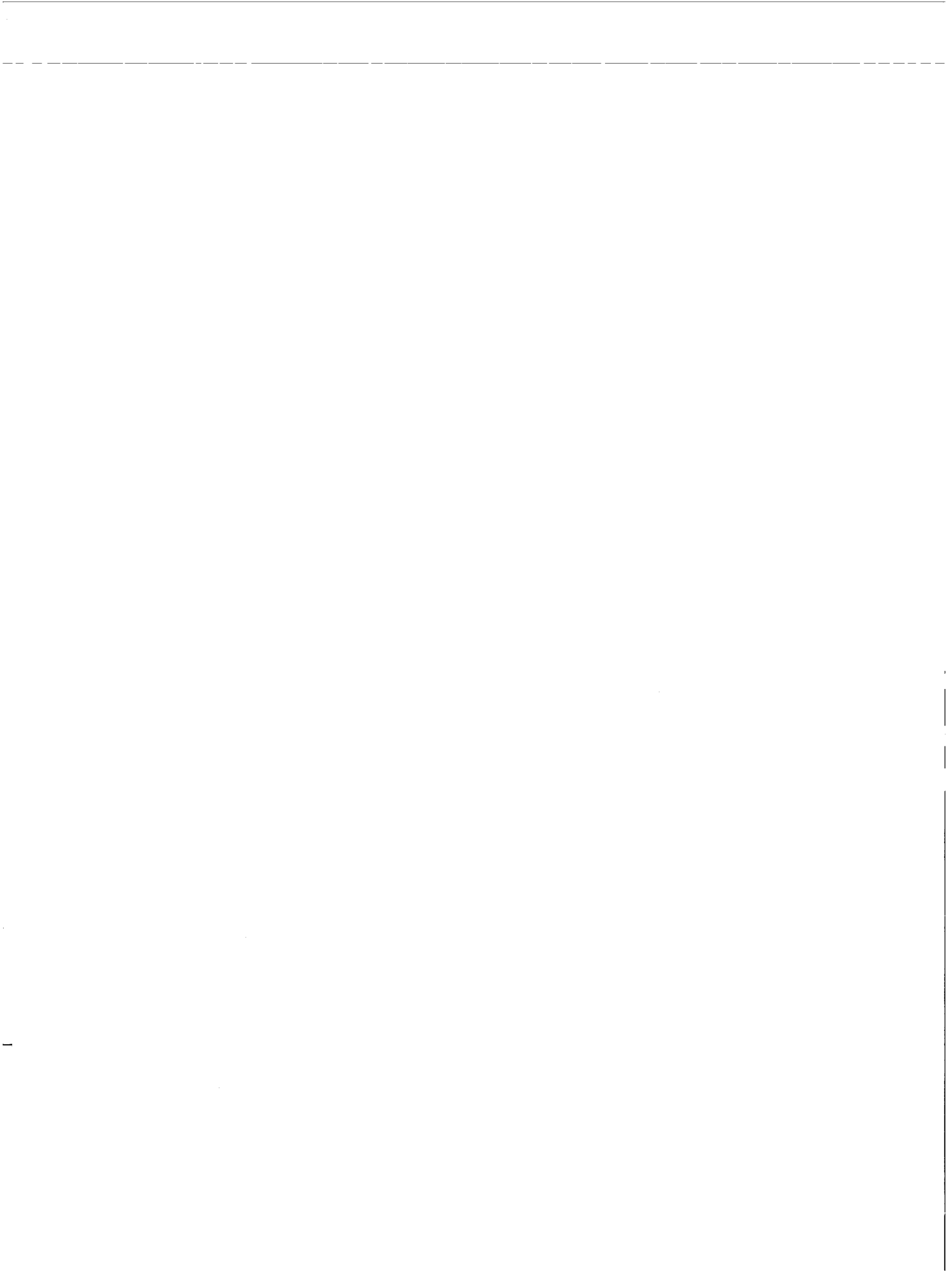
McMaster University
Hamilton, Ontario

TITLE: Automated Message Triage - A Proposal for Supervised
Semantic Classification of Messages

AUTHOR: Amir Tavasoli, B.Sc

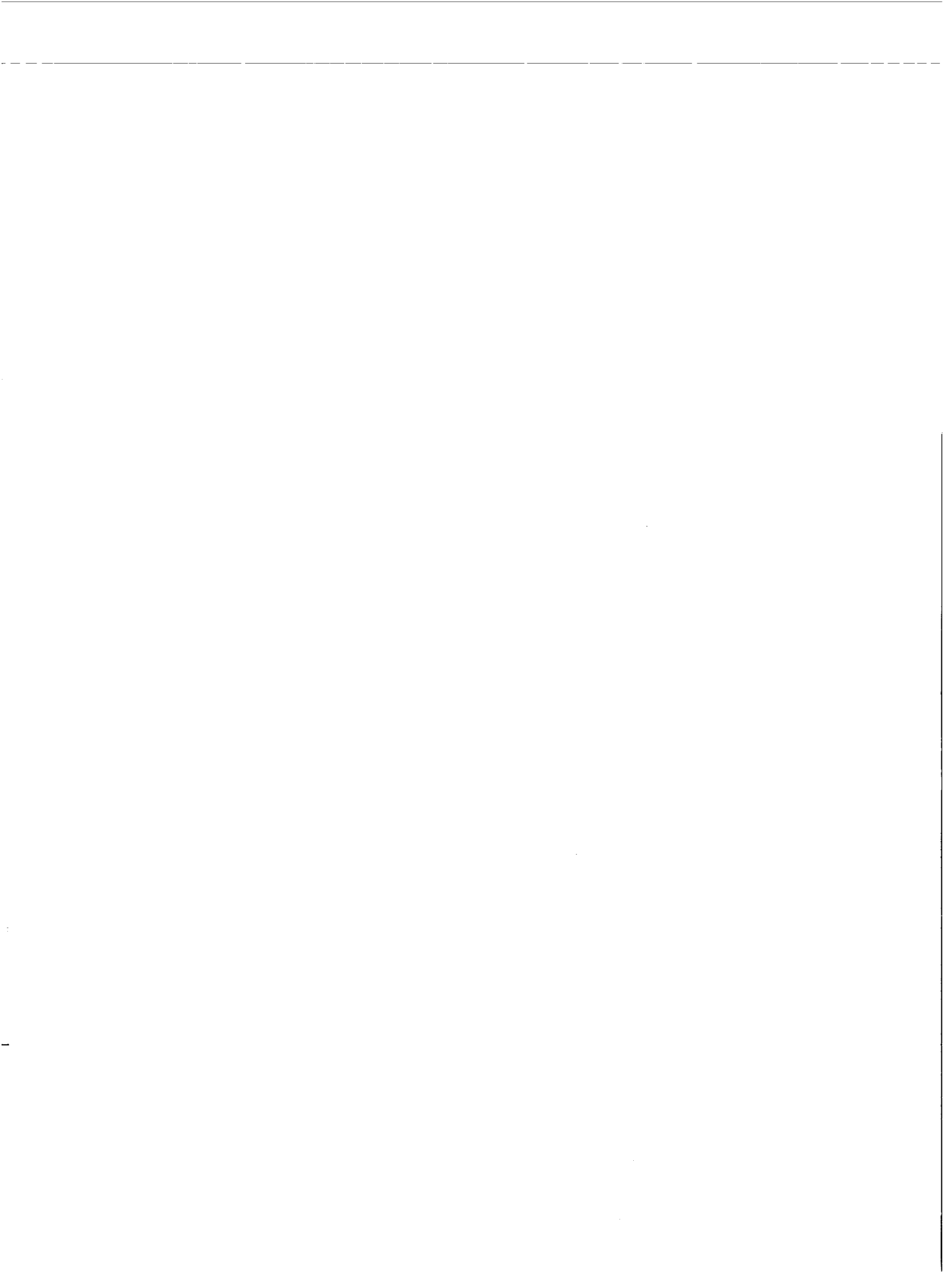
SUPERVISOR: Norm Archer, Ph.D., Professor Emeritus

NUMBER OF PAGES: xi, 127



Abstract

Classification or Categorization is a text mining technique in which the given text documents are classified into specified categories. There are several techniques for classifying messages, ranging from simple K Nearest Neighbors to complicated Support Vector Machines. These classifiers have proven to be effective in cases where the documents in each category do not have a great deal of overlap with other documents. Designing a classifier that is effective in environments where there is no way to avoid this overlap, like emails, text messages, or user opinions and comments, has remained a continuing challenge. This work is a proposal for a system that classifies such documents based on their content so they can be sorted by semantic significance. This has several applications in the real world, like triaging patient messages to physicians in the healthcare field or sorting user opinions on a product webpage. We have combined and tailored different classifiers to build a high performance classifier that supports this type of classification. The system has been tested and proven to have good performance with real-world user messages that were exchanged between patients and physicians during a hypertension prevention study.



*Dedicated to my parents, Fatemeh Khosh and Eskandar
Tavasoli*

Acknowledgments

The last two years have been the most challenging and yet amazing years of my life. I have experienced great experiences, found great friends and overcome several challenges. Overcoming these challenges would have been impossible without the help of my family, friends and colleagues. I would like to express my gratitude towards the following people for their great support.

I have to admit that without the support of my supervisor, Dr Norm Archer, it would have been impossible for me to accomplish my goals. I have always been motivated by his interest and energy in his teaching and research, and his absolute desire to share his wealth of knowledge with his students and colleagues and helping other human beings. I should say he is not only a great supervisor but also a marvelous human being. I feel really lucky and have learned great lessons in life because of being his student. His support and assistance and his dedication to guiding me during the past two years is greatly admired and appreciated.

I would also like to acknowledge my deep appreciation to my family who dedicated their lives to helping me, and it is impossible for me to compensate for what they have done for me. I am really thankful to have grown up in such a loving and supportive environment. A very special thanks to my true and beloved friends, who now live in the four corners of the world, and whose support and kindness have contributed greatly to my life's journey; specifically Samaneh Gheibi, Mohammad H. Koleini, Farbod Riahy, Afshin and Ramin Khezri, M. Reza Heydarian and Sepandar Sepehr.

I would like to express my acknowledgment to myBP Trial Team in Department of Family Medicine at McMaster University; specifically Dr. David Chan, Christine Rodriguez, and Lisa Dolovich. Their kind support in making the messages available and help in triaging the data is greatly appreciated.

I would also like to express my thanks to the Alias-i Company for developing the free open-source text mining product, LingPipe. Their sharing of the code and development details of this product is very much appreciated and has contributed greatly to this and many other works.

Table of Contents

ABSTRACT	III
ACKNOWLEDGMENTS	V
TABLE OF CONTENTS	VI
LIST OF TABLES	VIII
LIST OF FIGURES	IX
GLOSSARY	X
1. INTRODUCTION	1
2. REVIEW OF THE LITERATURE AND RESEARCH	4
2.1 TEXT MINING	4
2.1.1 <i>Text Mining Steps</i>	5
2.1.1.1 <i>Tokenization</i>	5
2.1.1.2 <i>Feature Reduction or Selection</i>	6
2.1.1.3 <i>Document Classification</i>	9
2.1.1.3.1 <i>K Nearest Neighbor</i>	9
2.1.1.3.2 <i>Naïve Bayes Classifier</i>	10
2.1.1.3.3 <i>Regression Methods</i>	11
2.1.1.3.4 <i>Neural Networks</i>	11
2.1.1.3.5 <i>Support Vector Machines</i>	11
2.1.1.3.6 <i>Decision Tree Classifiers</i>	12
2.1.1.3.7 <i>Bayesian Network Classifier</i>	12
2.1.1.3.8 <i>Decision Rule Classifier</i>	13
2.1.1.4 <i>Multiple Classification Combination</i>	13
2.1.1.5 <i>Classifier Performance</i>	16
2.2 SIMILAR SYSTEMS.....	17
2.3 APPLICATIONS.....	19
2.3.1 <i>eHealth</i>	19
2.3.2 <i>Opinion Mining</i>	21
3. AUTOMATED MESSAGE TRIAGE (AMT) SYSTEM	23
3.1 eHEALTH AS THE MAIN APPLICATION.....	24
4. UNDER THE HOOD	26
4.1 DATA GATHERING AND PREPARATION	26
4.1.1 MYBP STUDY.....	26
4.1.2 EXCHANGED MESSAGES.....	28
4.1.3 DATA CLEANING	29
4.2. CLASSIFICATION	33
4.2.1 CLASSIFIERS TESTED.....	35
4.2.1.1 <i>Language Model (LM) Classifier</i>	36
4.2.1.2 <i>Indo-European Tokenizer</i>	37
4.2.1.3 <i>Naïve Bayes Classifier</i>	38

4.2.1.4 <i>K Nearest Neighbor (Knn) Classifier</i>	39
4.2.1.5 <i>TF-IDF Classifier</i>	41
4.2.2 COMBINING CLASSIFIERS	43
4.2.2.1 <i>Training Multiple Classifiers</i>	43
4.2.2.2 <i>Simple Voting</i>	44
4.2.2.3 <i>Dynamic Classifier Selection</i>	46
4.2.2.4 <i>Adaptive Classifier Combination</i>	46
4.2.2.5 <i>New Proposed Combination Method</i>	47
4.2.3 COMPARISON OF CLASSIFICATION METHODS	48
4.2.3.1 <i>Benchmark Conditions</i>	49
4.2.3.2 <i>Classifier Confusion Matrices and Test Parameters</i>	49
4.2.3.2.1 <i>Language Model (LM) Classifier</i>	49
4.2.3.2.2 <i>Naïve Bayes Classifier</i>	51
4.2.3.2.3 <i>K Nearest Neighbor (Knn) Classifier</i>	51
4.2.3.2.4 <i>TF-IDF Classifier</i>	53
4.2.3.2.5 <i>Simple Voting Classifier</i>	53
4.2.3.2.6 <i>Dynamic Classifier Selection</i>	54
4.2.3.2.7 <i>Adaptive Classifier Combination</i>	55
4.2.3.2.8 <i>New Adaptive Classifier Combination</i>	57
4.2.3.3 <i>Choosing the best method</i>	59
4.3 CONNECTING THE DOTS	60
5. FUTURE WORK	61
BIBLIOGRAPHY	63
APPENDICES	70
APPENDIX I	70
APPENDIX II	72
APPENDIX III	74
APPENDIX IV	79
APPENDIX V	91
APPENDIX VI	125

List of Tables

<i>Table 1 - Spreadsheet Representation of Text Documents [3]</i>	6
<i>Table 2 - Thresholding Frequencies to Three Values</i>	6
<i>Table 3 - Decision Tree Structure [10]</i>	12
<i>Table 4 - Patient Characteristics Table [81]</i>	27
<i>Table 5 - MyBP Patient Message Categories</i>	30
<i>Table 6 - Triage Level Details</i>	31
<i>Table 7 - Number of Messages by Triage Level</i>	32
<i>Table 8 - Classifiers Used in This Work</i>	36
<i>Table 9 - Indo-European Tokenizer Tokens [93]</i>	37
<i>Table 10 - Language Model (LM) Confusion Matrix (n-gram size = 7)</i>	50
<i>Table 11 - Naive Bayes Confusion Matrix</i>	51
<i>Table 12 - Effects of DistanceFunction on Knn Classifier</i>	52
<i>Table 13 - Knn Confusion Matrix (k=2, Euclidean Distance)</i>	52
<i>Table 14 - TF-IDF Confusion Matrix</i>	53
<i>Table 15 - Simple Voting Classifier Confusion Matrix</i>	54
<i>Table 16 - Different Distance Functions for Calculating DCS Neighborhoods</i> <i>(with k=4)</i>	55
<i>Table 17 - Dynamic Classifier Selection (DCS) Confusion Matrix (k=4 and</i> <i>Euclidean Distance)</i>	55
<i>Table 18 - Different Distance Functions for Calculating ACC Neighborhoods</i> <i>(k=4)</i>	56
<i>Table 19 - Adaptive Classifier Selection (ACC) Confusion Matrix (k=4, Cosine</i> <i>Distance Measure)</i>	57
<i>Table 20 - New Adaptive Classifier Combination Confusion Matrix (k=9)</i>	58
<i>Table 21 - Classifier Comparison</i>	59
<i>Table 22 - Resolving Differences Between Message Triage Levels Chosen by Two</i> <i>Nurses</i>	85
<i>Table 23 - Messages Simulated to Overcome Lack of Level 0 and Level 1</i> <i>Messages</i>	90

List of Figures

<i>Figure 1 - Classification Process [3]</i>	9
<i>Figure 2 - Selection Model for Multiple Classifiers (adapted from [60])</i>	14
<i>Figure 3 - Combination Model for Multiple Classifiers [60]</i>	15
<i>Figure 4 - Newsblaster Customized News About Sports</i>	18
<i>Figure 5 - Screen Capture from Mozilla Thunderbird Spam Guard</i>	19
<i>Figure 6 - A Screenshot of the myOSCAR Personal Health Record System</i>	20
<i>Figure 7 - Automated Message Triage System</i>	23
<i>Figure 8 - myBP Study Phases [81]</i>	26
<i>Figure 9 - myBP Messaging System [81]</i>	28
<i>Figure 10 - Steps in Data Preparation</i>	33
<i>Figure 11 - Text Categorization: Training a Model and Classifying a New Document [1]</i>	34
<i>Figure 12 - Classification Process of Naive Bayes Classifier [1]</i>	39
<i>Figure 13 - Knn Classifier [1]</i>	40
<i>Figure 14 - TF-IDF Classifier Classification Process [1]</i>	42
<i>Figure 15 - Simple Voting Classifier Combination with Modification</i>	45
<i>Figure 16 - n-gram Size for LM Classifier Performance</i>	50
<i>Figure 17 - k Size to Knn Classifier Performance</i>	51
<i>Figure 18 - k Dynamic Classifier Selection (DCS) Classifier Performance</i>	54
<i>Figure 19 - k Size to Adaptive Classifier Combination Classifier (ACC) Performance</i>	56
<i>Figure 20 - k Size to New Adaptive Classifier Combination Classifier Performance</i>	57
<i>Figure 21 - Automated Message Triage Overall Design</i>	60
<i>Figure 22 - GATE Opening Screen</i>	74
<i>Figure 23 - GATE with Document Corpus</i>	75
<i>Figure 24 - Running ANNIE in GATE</i>	76
<i>Figure 25 - GATE Finding Private Information</i>	77

Glossary

ACC: Adaptive Classifier Combination – method for combining classifiers.

AMT: Automated Message Triage – the system proposed in this work.

ANNIE: A Nearly-New Information Extraction System - a component for extracting information - for instance tagging data in GATE.

API: Application Programming Interface – an interface that is provided by a software package in order to interrelate with other software applications.

Cerr: Critical Error – a type of error defined in 4.2.3 Comparison of Classification Methods.

DAG: Directed Acyclic Graph – type of graph that has no cycles.

DBP: Diastolic Blood Pressure – minimum blood pressure.

DCS: Dynamic Classifier Selection – method for combining classifiers.

DNF: Disjunctive Normal Form – form of logical formula which consists of disjunction of conjunctive logical clauses.

EMR: Electronic Medical Record system – an EMR system that assists healthcare professionals in managing health related information electronically.

ePHR: electronic Patient Health Record system – a system that lets patients update, access, and control their health information online.

erate: Error Rate – a type of error

GA: Genetic Algorithm – a machine-learning algorithm.

GATE: General Architecture for Text Engineering - a text mining application developed at the University of Sheffield in England.

IR: Information Retrieval

KDD: Knowledge Discovery from Data – a synonym for Data Mining.



Knn Classifier: K nearest neighbor Classifier – a classifier used for categorizing text documents.

LM Classifier: Language Model Classifier – a classifier that is used in text mining.

myBP: a study that was conducted by the Family Medicine Department at McMaster University to measure the effects when hypertension patients used an ePHR to help manage their illness.

myOSCAR: an ePHR developed by the Family Medicine Department at McMaster University.

NN: Neural Network – a machine learning method.

OSCAR: Open Source Clinical Applications & Resources – an EMR system that was developed by the Family Medicine Department at McMaster University.

RSS: Really Simple Syndication – xml-based web news feeds.

SBP: Systolic Blood Pressure – maximum blood pressure.

SE: Standard Error – a type of error.

SVM: Support Vector Machine – a machine learning method.

TF-IDF: Term Frequency - Inverse Document Frequency – a very useful method that is used in text mining and is described in 2.1.1.2 Feature Reduction or Selection.

WEKA: Waikato Environment for Knowledge Analysis – a text mining application that was developed by the Department of Computer Science in University of Waikato in New Zealand.

1. Introduction

The introduction of the World Wide Web not only gave rise to a new era in mankind's history; it very quickly resulted in several new challenges. One of the most important was an explosion of information generally available online to everyone with access to the Web [1]. To overcome this challenge, techniques known as knowledge discovery or data mining have emerged [2]. A major issue with this vast ocean of information is its unorganized nature. Organizing this gigantic amount of information would be difficult, time consuming, and probably an impossible obstacle to overcome. One approach to this problem is text mining [3] which goes one step further, claiming to be able to obtain desired knowledge from the Web without first organizing it. This has resulted in a significant amount of attention to research in text mining, and a major trend in the use of this innovative approach [1].

A need to have a system that is able to sort text documents semantically can be found in several areas, ranging from healthcare to opinion mining [4]. With such a system, example applications include: sorting messages sent to a medical doctor, based on their significance; sorting reviews left by users on a product webpage, based on how favorable the reviews are; sorting office emails based on their importance or urgency; or sorting comments in a blog based on how interesting the comments are. This research work is an attempt to tackle the semantic sorting of text documents, using an innovative approach to text mining.

The inspiration for this work came mainly from the healthcare arena and the application of electronic Personal Health Record (ePHR) systems. An ePHR system is a system that enables patients to access and manage their own health information online. Studies have suggested that the use of ePHRs has a positive effect on patient doctor-patient communication [5]. Most ePHR systems provide a method for patients and physicians to communicate with each other through secure messaging [6]. When an ongoing connection between patient and physician is provided, this changes encounters from episodic to continuous, and reduces the time needed to address problems that may arise. Unfortunately, this increased rate of communication can increase workloads on already overloaded physicians, resulting in an insuperable barrier to the use of this potentially valuable tool. We were inspired to look at this problem by the potential of semantically triaged messages, especially in regions that lack ready access to healthcare professionals, such as developing countries [7].

There are several techniques that can be used in text mining to manage this challenge, ranging from handcrafted to machine learning methods. Due to the complex nature of the healthcare environment, predictive machine learning methods are the best suited to mining unstructured information [3]. In addition, because of the sensitive nature of messages relating to medical care, a high level of precision is needed to ensure that false negative handling of incoming urgent messages is virtually eliminated. To reach this level of precision, this research included the selection and use of certain text mining methods that have been especially tweaked to fit in this environment until they demonstrated robust performance. Obviously these tweaks can be generalized and are useful in any environment where semantic triage of text documents is required.

To simulate a real word situation and ensure optimal performance of our system we used a test bank of real world messages from patients and caregivers that were exchanged in the myBP trial. MyBP was a project undertaken recently by the McMaster Department of Family Medicine that was designed specifically to examine the effects of using an ePHR on patient self management of hypertension. In this trial an adaptation of myOSCAR , a secure open-source ePHR system that was developed at McMaster University [8], was used to help monitor and manage patients with hypertension. During the trial, experienced professionals managed all incoming messages from patients through a triage process. In our research we examined the use of text mining approaches to automate the triage process so that less urgent messages could be handled automatically while ensuring that truly urgent messages are forwarded to on-call professionals for rapid response. This automated triage process can result in a reduction in the workload on professionals supporting self management of healthcare. Moreover there are many other potential applications of automated message triage, some of which have already been mentioned.

To develop and test techniques for automated triage, the system that was designed was trained with part of a test bank of existing messages, and then tested before actual use with the remainder of the test bank to ensure reliability. The test messages were each assigned a priority level in advance by a triage professional, in order to test the ability of the automated system to properly classify the test messages. Before using the myBP messages, they were completely anonymized by deleting their headers and any other personal identifying information remaining in the text, to ensure the privacy of patients and healthcare professionals. This work was approved by the McMaster Research Ethics Board.

The automated system can also be used to triage text messages that are sent from patient's cell phones. It can also be generalized such that it will be able to respond to certain classes of simple messages (e.g. requests for background information) that are delivered to the system. By managing less important or urgent messages, this reduces the communication burden on health professionals. Future research could be done that would continue developing better performance and precision from the system. This system, given the name Automated Message Triage (AMT) system has shown good performance in the healthcare field where the messages are short and there is significant overlap between message triage levels. The adaptation of AMT to other less critical environments such as blogs or product reviews appears to have much future promise.

2. Review of the Literature and Research

2.1 Text Mining

Due to rapid growth in the amount of online information in recent years, especially after the emergence of the Web, as the father of Information Retrieval (IR) Gerald Salton predicted, we are being forced to mine the information that we need out of the huge amount of information available [1]. Data Mining, which is “*extracting or mining knowledge from large amounts of data*”, has received a great deal of attention in recent years due to the changes described [2]. Actually the term “data mining” is a slight misnomer because we often refer to extracting metal ore from rock as metal mining and not rock mining. However, because of length of the term “knowledge mining from data” and the loss of emphasis on the large proportions of data in term “knowledge mining”, the term “data mining” has become common in the literature and in industry [2]. Another synonym for the term Data Mining is Knowledge Discovery from Data or KDD which is also commonly used [2]. Here, the word “*knowledge*” means an interesting pattern in a domain of relevance [9].

Data mining almost always requires highly structured data, which takes an enormous effort to prepare [3]. Moreover, there are some cases where we need to make real time decisions based on new data, and there is no time to prepare the data for the process. This is the case in Web searches or guarding against spam, etc. Our interest in this research is in Text Mining, the process of extracting interesting patterns from document collections [10].

Here are the main problems that can be solved using text mining:

1 – Document Classification

This is also known as text categorization, and is the most common use of text mining. The problem to be solved is: given a set of categories, classes or subjects, then automatically decide to which class a new document belongs [10]. This problem is the main focus of this thesis.

2 – Information Retrieval

This is also known as information extraction. Information retrieval can be done in several ways. One of the most common is to, when given a query, search through a big corpus of documents and find the documents that match the given query. This method is used every day in Web search engines [3]. Another approach is tagging documents and finding useful information in them, such as entities, attributes, facts or events. For instance, this can result

in finding names, the size of peoples' feet, the names of a company's employees, names of persons who were involved in a car accident in a given corpus of text documents, etc. [10]. This approach is widely used for different purposes, from stock market prediction to tracking criminal actions on the Web. It can be done using GATE software (mentioned in the following chapters) [11]. The information extracted from a large corpus of text documents can be shown visually with various methods to make the results readily understandable [10].

3 – Clustering and Organizing Documents

The objective here is to automatically find the categories (classes or subjects) in an unstructured corpus of documents. Each category should contain similar documents and these are determined by the data itself [3]. Clustering is typically used for data analysis in pattern recognition, image segmentation, etc. [10]

2.1.1 Text Mining Steps

Text mining steps are quite analogous to data mining steps, but in text mining we need to take the highly unstructured nature of our environment into consideration [3]. Each of the applications mentioned has specialized steps. For example, steps that are needed for visualizing information are different from steps taken in clustering. In the following we describe the steps needed in document classification.

2.1.1.1 Tokenization

The initial procedure is converting the text data into a format that is understandable for the mining process, which requires numerical data. The first issue is to convert character sequences into words or (in other words) tokens. To do so requires an awareness of the language structure, because certain characters are sometimes token delimiters and sometimes they have other uses. For instance, a period can be part of an email address or the ending of an sentence. There are good precise algorithms for tokenizing English language text [3].

Data Representation

After reading tokens, a good representation is needed that can be saved easily in a data structure for storing or processing in main memory. One of the most common data models used is called the Spreadsheet Model. An example of a spreadsheet that can represent a text document (see Table 1), is a record of the word frequencies in a set of documents [3].

	CN Tower	Celebration	Toronto
Document 1	1	0	5
Document 2	0	2	1
Document 3	1	0	1
Document 4	2	4	0

Table 1 - Spreadsheet Representation of Text Documents [3]

This representation has been used in [12-17] according to [18]. There is another representation of a document, in which a Boolean spreadsheet is used only to record whether or not a word appears in a document. This approach has been used in [17, 19-22] according to [18]. There are some combinations of methods like representing 'Thresholding Frequencies'. For a Thresholding Frequency with three values for example, we use 0, 1 and 2 numbers in a spreadsheet according to the pattern in Table 2. This scheme has the benefit of greater performance when recording frequencies [3].

Number	When to use
0	Document does not contain the word
1	The word occurred only one time
2	The word occurred more than one time

Table 2 - Thresholding Frequencies to Three Values

Another method of data representation is to use certain additional information in building a spreadsheet [18]. One well-known method is the use of n-grams, which are sequences of n words (or characters) instead of one word. In this method groups of words (or characters) that occur frequently (e.g. 'United States of America') can be treated as one word, thus reducing the number of features. There is a good algorithm for doing this [18, 23]. This method is also known as a multiword feature [3].

All the data representations mentioned so far are known as 'bag-of-words' models. Another method that can be used in to build a spreadsheet would be the word position in the text, resulting in a model that is not a bag of words [18, 20].

2.1.1.2 Feature Reduction or Selection

After identifying tokens in the text, it is better to reduce the number of tokens and keep only the significant and relevant tokens in most cases. This technique is called feature reduction. Another technique is feature selection, which is recoding all the features and selecting only the related ones in the

final vector, which is the vector that is made from set of features and used for classification. The approach selected is based on the situation. Feature reduction or selection has been proven to have a positive effect on most classification results, due to the omission of unwanted or insignificant features [3] i.e. dimensions of our final vector. There are several widely used methods for feature reduction, as follows.

One simple and common feature reduction method is lemmatization or stemming. In lemmatization all the words of the same type are considered as one entry. These words can be words of the same root, words that are synonyms of each other, etc. so they are all considered to be one entry [3]. Another very common approach for reducing features is removing stop words like a, it, and, etc. from the list of features [3].

Another method for reducing the number of features is to remove words that are the most frequent. But just removing the words that are most frequent can be risky in most cases because these words can be rare in other documents, making the words valuable. To overcome this deficiency different methods have been suggested, with a well-known one being TF-IDF which stands for Term Frequency - Inverse Document Frequency. In the TF-IDF method each token is assigned a weight, which is calculated as:

$$\begin{aligned} \text{weight}(\text{token}) &= \text{tf}(\text{token}) * \text{idf}(\text{token}) \\ \text{tf}(\text{token}) &= \text{token frequency in the document} \\ \text{idf}(\text{token}) &= \log\left(\frac{N}{\text{df}(\text{token})}\right) \\ \text{df}(\text{token}) &= \text{number of documents the word is used in} \\ N &= \text{total number of documents} \end{aligned}$$

In TF-IDF words are weighted based on their importance, so the only words chosen are the words that have greater importance (thus greater weight.) There are different versions of TF-IDF; for instance, weighting different words from different parts of a document differently or having different weights for different categories [3] [24]. Salton and Buckley describe several alternative versions of TF-IDF and their usage [25]. Another interesting alternative version of TF-IDF has been mentioned in [18, 26], where tokens are weighted using “mutual information between word occurrence and class value”.

Other than the TF-IDF weighting scheme, there are certain other weighting algorithms that can be used to reduce the number of features. For example, the χ^2 measure which calculates the dependence between the

feature and the class, has proven to be very effective in dimension reduction of vectors [10, 27]. H. Schütze and et al. recommend using a best-term χ^2 measure that is calculated locally [26, 28, 29]. Another weighting scheme is known as Information Gain (IG). For each token a probability measure called IG is calculated. This represents the information gathered from a prediction of its lack or occurrence in the document containing the token in this category [10].

Machine Learning algorithms can also be used in feature selection. Lutu and Engelbrecht suggest using heuristic search for finding highly related features in a document [30]. There are several other described in a patent by Weston et al [31]. There are several feature selection algorithms that are mostly used in image recognition but can be generalized for text feature sets. Good examples are sequential forward floating selection (SFFS) or “plus 1-take away r” selection methods [26, 32].

Another way to reduce the number of features in the final vector is to map it into a more useful lower dimension. A linear projection method can be used to mirror the vector into a vector combination from the first vector. This method is known as feature extraction [26]. Related multiword groups can be addressed using the n-gram data model, or words can be grouped that are semantically related, using supervised or unsupervised clustering. Using these word groups can mirror the current vector to a vector with a lower dimension [10].

Research has also shown that using background information of the categories can result in certain improvements [10, 33, 34]. On the other hand, unsupervised learning has proven to have weak results [10, 26, 35, 36].

Another approach to feature extraction is latent semantic indexing (LSI). LSI uses Singular Value Decomposition (SVD) to find relationships between tokens in the document. The dimensions of a data model spreadsheet generated using the SVD principle will be reduced. Research has shown that, like the previous case using background information, this can produce some improvements in categorization [10].

Having completed tokenization and feature reduction or selection, the document is ready to be processed. The methods used for classification will be covered in the following.

2.1.1.3 Document Classification

Classifying documents requires as an input the document features. The classification process requires that the classifier be trained to recognize pre-defined categories so it can predict to which category an unknown (or new) document belongs [3].

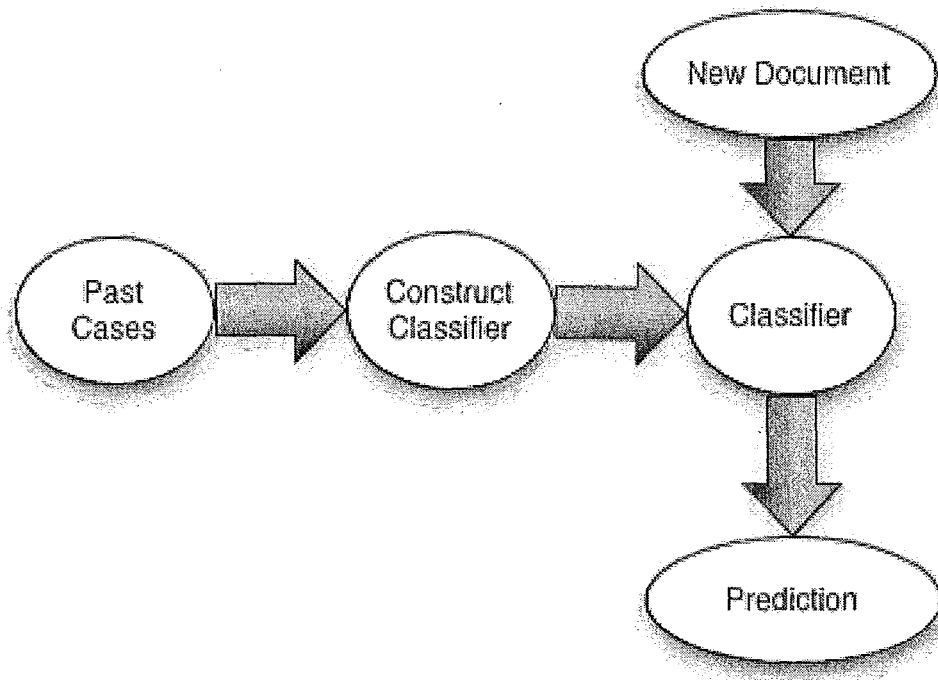


Figure 1 - Classification Process [3]

The following is a brief description of some well-known classification algorithms and some of their interesting alternatives.

2.1.1.3.1 K Nearest Neighbor

K Nearest Neighbor methods are based on similarities between documents. In the training phase it is necessary only to save the training vectors and to which category they belong. Analyzing a new document involves finding its k nearest neighbors. The result is a prediction that the document belongs to the most frequent category it shares with its neighbors. A similarity measure is used to find the document's k nearest neighbors. The most common neighbor similarity measure is Euclidian Distance between vectors [3]. Other similarity measures include Cosine Distance between vectors or the angle

between two vectors, which Qian, Sural and et al have proven to be no worse than the Euclidian Distance measure [37]. Another interesting distance measure is Taxicab Distance, which is the distance between vectors measured with Taxicab Geometry. The Taxicab Distance is a projection of the distance on line segments between the two vectors i.e. “the way that a Taxicab would drive between two points” [38]. Taxicab Distance is also known as Manhattan Distance, City Block Distance, and L_1 Norm Distance [39]. Suppose $\vec{Q} = (q_1, q_2, \dots, q_n)$ is a vector of numerical features and $\vec{V} = (v_1, v_2, \dots, v_n)$ is also a vector composed of numerical features. Then the taxicab distance between these vectors is calculated as:

$$d_{taxicab}(\vec{Q}, \vec{V}) = \sum_i |q_i - v_i|$$

Another interesting similarity measure is Minkowski Distance, which is the generalized version of the Euclidian Distance [39]. The Minkowski Distance is calculated as follows:

$$d_{Minkowski}(\vec{Q}, \vec{V}) = \left(\sum_i |q_i - v_i|^p \right)^{1/p}$$

P in the equation for Minkowski Distance is known as the order of the Minkowski Distance. For instance if $p=2$ then Minkowski Distance will be the same as Euclidean Distance.

Different orders of Minkowski Distance have been proven to have good performance in document classification [40].

2.1.1.3.2 Naïve Bayes Classifier

The Naïve Bayes Classifier is based on Bayes' theorem of probability. Given a new document the probability of this document belonging to a certain category is calculated using Bayes' theorem. For each class we calculate the probability that document d belongs to class c [10].

$$\Pr(c | d) = \frac{\Pr(d | c) \cdot \Pr(c)}{\Pr(d)}$$

$\Pr(d)$ is a constant so this is not calculated. However for $\Pr(d|c)$ it is necessary to assume that tokens in document d are independent [10].

$$\Pr(d | c) = \prod_i \Pr(token_i | c)$$

Because of this weak assumption, this method is known as Naïve Bayes [10]. Interestingly enough this “naïve” assumption is known to perform robustly [41, 42].

2.1.1.3.3 Regression Methods

Regression is a statistical method used for predicting new values of a function from its known values. Regression models can be used for text classification. Yang and Chute suggested using the linear least-squares fit (LLSF) in text classification [10, 43]. Another regression categorization method that has performed well is Bayesian Logistic Regression [44].

2.1.1.3.4 Neural Networks

Neural Networks (NN) can also be used to perform text categorization. The first step is designing an NN that can be trained using data from known documents and their classification, and then to use the trained network to classify new documents. For text classification mostly feed-forward back-propagation neural networks are used, in which weights are determined by propagating error through the networks [10]. Studies of these types of networks have shown that a perceptron (one-layer neural network) performs as well as NNs with more layers (i.e. hidden layers) [10, 45, 46]. However, combining multiple neural networks in a hierarchical structure, (hierarchical neural networks), has been shown to be very effective [47].

2.1.1.3.5 Support Vector Machines

A Binary Support Vector Machine is simply an optimal hyper-plane in feature space, a high dimensional vector space which separates vectors that belong to a category and those that do not belong to that category. These binary support vector machines find a certain hyper-plane by using a small number of instances called Support Vectors, while the rest of the training instances are mostly neglected [10]. Huang and Wang suggest using Genetic Algorithms (GA) for finding parameters of SVM in order to increase performance [48]. They also recommend using a fuzzy SVM for multiclass classification [49].

2.1.1.3.6 Decision Tree Classifiers

A Decision Tree is a tree which has a structure as outlined in Table 3.

Node Type	Representation
Internal nodes	Features
Edges between nodes	Features weight
Leaves	Classes

Table 3 – Decision Tree Structure [10]

For classifying a document, this process starts from the root and chooses the edge that satisfies the given document condition, continuing on in this manner until a leaf is reached. The document is classified according to the final leaf's label. Most Decision Tree classifiers are used for binary classifications, and are therefore binary trees [10].

There are several algorithms for Decision Tree learning. However, most of them (e.g. ID3, C4.5, C5 [26] and CART) follow a recursive procedure, which picks a feature and divides the training samples into those that have that feature and those that do not, until a leaf is reached. The final step occurs when the training sample contains only one class. Choosing the right feature to continue is mostly done through a measure like information gain. One of the disadvantages of decision trees is that they can overfit to a certain category, so most of these algorithms use pruning methods to avoid overfitting. The main advantage of using decision trees is that they are readable by humans [10].

2.1.1.3.7 Bayesian Network Classifier

Bayesian Networks are well-known statistical structures. They are Directed Acyclic Graphs (DAG) that are labeled with probability potentials. Each node in a tree represents a feature, and edges represent their conditional probabilities [50]. When given a new document, the algorithm uses the Bayes chain rule to calculate the probability that the document belongs to a certain category.

There are several learning algorithms for Bayesian Networks, which use complex statistical approaches to build the graphs and assign probabilities to them [50]. Klopotek and Woch suggest that tree-like Bayesian Networks perform well in large vector spaces [51]. Sanchez-Graillet and Poesio suggest a new method for building a Bayesian Network from text. It finds the causal relationship between words (tokens) in a text document

[52]. Sandeep [53] presents some very good work on the use of Bayesian Networks in text mining.

2.1.1.3.8 Decision Rule Classifier

Decision Rules are simple Disjunctive Normal Form (DNF)- like rules that are built from a document using inductive rule learning. If a document has n features, each represented by d_i that belong to class c , the first step is making the initial rule:

$$d_1 \wedge d_2 \wedge \dots \wedge d_n \rightarrow c$$

The next steps involve optimizing these base rules, followed by pruning to avoid overfitting the rules [10]. One interesting rule-based classifier system is RIPPER, which stands for “Repeated Incremental Pruning to Produce Error Reduction”. Ripper has an interesting capability to improve performance towards either precision or recall [10, 20, 54, 55].

2.1.1.4 Multiple Classification Combination

As in the real world where a committee of experts can usually make better decisions than just a single person, a combination of classifiers has the potential for better precision than just one classifier. There are several methods for combining classifiers. Most of these methods require that the classifiers that are used differ significantly either in their data representation or in their learning methods [10].

There are two general approaches to classifier combination. The first method is *bagging*. In bagging, each classifier learns its way through the training data separately and in the end the classifier results get combined. The second way is *boosting*. In boosting, each classifier that learns from the training data reweights the training data and makes it ready for the next classifier to get trained using these revised weights [10]. Our work is focused on the use of the bagging method so bagging methods are reviewed extensively below.

The first and most obvious way to operate a bagging classifier is through voting. In this method the new document is classified to the group for which most classifiers voted [26]. There are variations to this method that classify the document to a certain class when more than half of the classifiers have voted that way [10]. Hansen and Salamon claimed that simple voting

can be very effective in certain situations [56, 57]. Another approach is to use a linear combination of classifier in which we assign a weight to each classifier and weight the results using a linear equation. The final decision is to choose either the best one or choose the best category based on that equation [10].

Another method for combining classifiers is selecting the best classifiers among the set of classifiers. This method is known as Selection, and is demonstrated in general in *Figure 2*. A well known selection algorithm is Dynamic Classifier Selection (DCS). Using DCS the k -nearest-neighbors of the given new document are used to determine which classifier does a better job in that local neighborhood and that classifier is then chosen to classify the given document [26]. As mentioned in 2.1.1.3.1 there are several ways for finding a neighborhood, so this method varies when different neighborhood methods are used. Another variation for DCS is that we can weight a neighborhood using cosine similarity [26, 56]. Giacinto and Roli represent a “theoretical framework for DCS” [58]. They also represent two variations for DCS. The first one is called “a priori selection” in which a portion of training data called the “validation data” is used to find the local neighborhood. This part of the data is extracted from training data and is not used for training. Another interesting variation which they propose is “a posteriori selection” in which the new document is classified before doing the selection and the results are used to find an appropriate neighborhood [58]. DCS has proven to have good performance in image classification [59]. Adaptive Classifier Selection is another appealing variation of DCS for combining classifiers. In this method, after the first step the classifiers that have a low certainty in the neighborhood of the new document are removed. In the second step the classifier in which there is the greatest certainty is chosen [56].

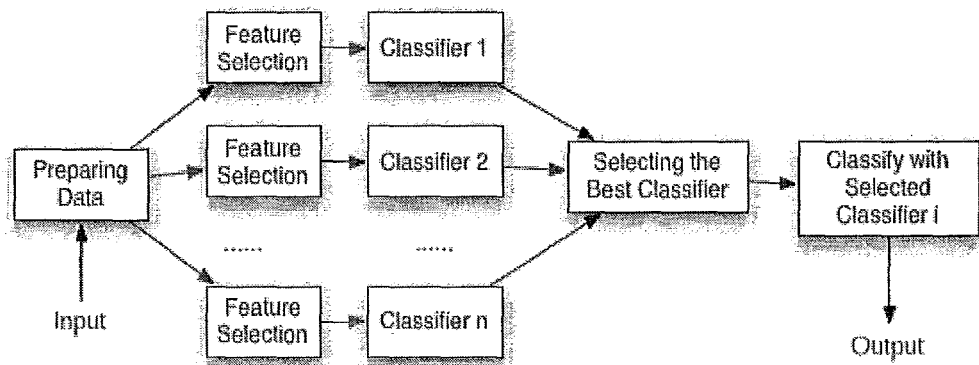


Figure 2 - Selection Model for Multiple Classifiers (adapted from [60])

Another approach for combining classifiers is called *Combination*, demonstrated in general in *Figure 3*. This involves selecting the best category for the new document based on the set of classifier results. A very strong method for Combination is known as Adaptive Classifier Combination (ACC). The first step in ACC is to find the neighborhood to which the given document belongs. The second step is finding the set of classes that the neighbors belong to. Then the local probability of each member of that set is found, in order to locate the class as the best representation for that neighborhood. That class is then the final choice [26]. ACC has proven to perform well and smoothly [26]. In the current research we obtained good results using ACC in a situation where even the classifier results were not very divergent. This further proves the strength of this technique.

Larkey and Croft suggest another interesting ranking combination algorithm for text documents in which the rank of each category is calculated using a scoring algorithm based on the combinational score of the k nearest neighbor algorithm and some other algorithm. The document is then categorized into the category with the highest score. They claim that this classifier works well [61]. Another interesting combination algorithm is due to the work by Zhang, Cheng and Ma. They suggest using a self-adaptive weighting combination method to combine classifiers using the weights [60].

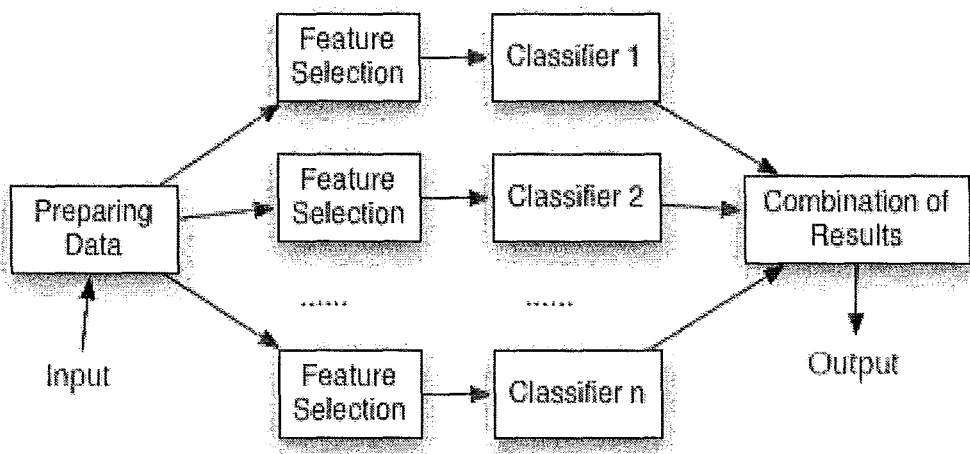


Figure 3 - Combination Model for Multiple Classifiers [60]

There are other combination methods like behavioral based selection [62] or combination [63], in which a combination is based on the behavior of classifiers for different categories. Yet another method involves using

maximum entropy to combine classifiers [60, 64]. One final technique is *adaBoost*, a boosting algorithm which uses an SVM-like concept to boost multiple classifiers [10].

...

2.1.1.5 Classifier Performance

Statistical measures are needed to compare classifiers. One measure is error rate, which is helpful since the intent of the study is to find the lowest possible classification error rates. Two error rate functions are [3]:

Error rate [3]:

$$erate = \frac{\text{number_of_errors}}{\text{number_of_documents}}$$

Standard Error [3]:

$$SE = \sqrt{\frac{erate \times (1 - erate)}{\text{number_of_documents}}}$$

Obviously these measures are too general and better measures are needed to evaluate text mining applications. Three ratios have been found to be very effective in analyzing text mining classifiers; precision, recall and F-measure [3].

Precision [3]:

$$precision = \frac{\text{num_of_correct_positive_predictions}}{\text{num_of_positive_predictions}}$$

Recall [3]:

$$recall = \frac{\text{num_of_correct_positive_predictions}}{\text{num_of_positive_class_documents}}$$

F-measure [3]:

$$F = \frac{2}{precision^{-1} + recall^{-1}}$$

Here, Precision is the percentage of documents that are correctly categorized. Recall is the percentage of all documents that are recognized for a certain category relative to all the documents in that category. The F-measure is the “harmonic mean of precision and recall” [3]. The F-measure is useful especially for classifiers that can change their precision or recall when parameters are adjusted; therefore the harmonic mean of these two ratios is the appropriate representation for classifier performance [10]. A very extensive evaluation method for comparing different classifiers is presented in Yang [65]. Yang describes several pitfalls that may result from using these measures, and suggests methods for avoiding them.

2.2 Similar Systems

There has been much research done in the text mining categorization area and many systems have been proposed. Here we will mention a few systems that are related or relatively close in concept and design to our proposed system.

One system that our work was inspired by is a classification system proposed by Li and Jian [26], a multi-classifier system for finding similar Web pages in a Web directory. Their system is claimed to perform well in spite of the fact that documents in different categories had many attributes in common, making the classification task a big challenge [26]. The work addressed in the current research follows a similar approach to classification but in a different environment. The challenge of the current problem is in the commonality of words, the sensitive environment, the need to change categories dynamically, and many other features that make it more difficult. However, the approach adopted in the current research was able to get better results than the proposed classifiers in the Li and Jian work.

Another interesting system is the “Personal Web Watcher” that suggests interesting Web pages to the user, whose interests it has already learned. It can also suggest interesting hyperlinks to a user who is searching for a key word on the Web. The system is basically a text mining classifier, in which all the text mining steps have been customized for its specific usage [18].

A fascinating text mining system is *Newsblaster* [3, 66]. *Newsblaster* is like an automatically generated newspaper that builds itself from contents that are available on the Web. This system, which was developed at Columbia University, crawls Web sites for news and then clusters them into different

categories. Then it classifies the categories it finds into the six most important categories. It summarizes the news and builds the output into a GUI that is published on the Web (See *Figure 4*). Newsblaster is a combination of several text mining techniques, including summarization, clustering, classification and several others. Its system overview and design is described by McKeown et al [66]. The classification methodology used in this work is a Naïve Bayes-like classifier that uses weights to classify the documents [67]. At the time of this writing it is accessible at <http://newsblaster.cs.columbia.edu/>. Google News (<http://news.google.com>) is a somewhat similar system that has been commercialized.

The screenshot shows a web browser window displaying the Columbia Newsblaster Sports News page. The browser's address bar shows the URL: <http://newsblaster.cs.columbia.edu/summaries/2010-08-10-04-26-37-sports.html>. The page title is "Columbia Newsblaster" and the subtitle is "Summarizing all the news on the Web". The date is "Tuesday, August 10, 2010" and the last update is "4:28 AM EST".

The page is organized into several sections:

- Search for:** A search bar with a "Go" button and a "Offline summarization" link.
- Navigation:** Links for "U.S.", "World", "Finance", "Sports", "Entertainment", and "Sports". There are also links for "View Today's Images", "View Archive", "About Newsblaster", "About Today's run", "Newsblaster in Press", and "Academic Papers".
- Article Sources:** A list of sources with the number of articles from each:
 - washingtonpost.com (117 articles)
 - usatoday.com (98 articles)
 - msnbc.msn.com (110 articles)
 - cbsa (30 articles)
 - signals.com (26 articles)
 - cn.com (26 articles)
- Sports Section:**
 - Orioles Third Baseman Melvin Mora Had Shoulder Surgery a Month After Season (Sports, 11 articles) [UPDATE]**: Includes a photo of a baseball player and a summary of the injury and surgery.
 - Mariners players deal with manager Don Wakamatsu's firing (Sports, 11 articles)**: Summary of the firing of manager Don Wakamatsu.
 - Washington Capitals Defenseman Brian Pothier Works on Returning to Form (Sports, 7 articles) [UPDATE]**: Summary of Brian Pothier's performance.
 - Cowboys beat Bengals in Hall game (Sports, 7 articles) [UPDATE]**: Summary of the game between the Cowboys and Bengals.
 - Thome HR fuels rally as Twins edge Tribe :: CHICAGO SUN-TIMES :: Baseball (Sports, 7 articles) [UPDATE]**: Summary of a game between the Twins and the Tribe.

Figure 4 - Newsblaster Customized News About Sports

Another type of system that has the same characteristics as the system used in the current research is email spam guards. In email spam guards the goal is to classify email into two categories: spam and not spam. Several email spam guards have been implemented and used commercially. Conferences are held each year that are dedicated to this specific issue; e.g. the *Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference* (formerly known as the *Conference on Email and Anti-Spam* or CEAS) [68]. A

well-known and effective open-source spam guard is Mozilla email filtering which has been implemented in Mozilla's recently introduced product *Mozilla Thunderbird Email Client* (see *Figure 5*). This classifier is trained as it is used by giving it feedback about which messages are spam and which are not [69]. It uses the Bayesian filtering algorithm to filter spam messages [3, 70]. The difference between an email spam guard and the present research is that our system needs to go further by evaluating the importance of each email so emails can be sorted, based upon their significance to the user.

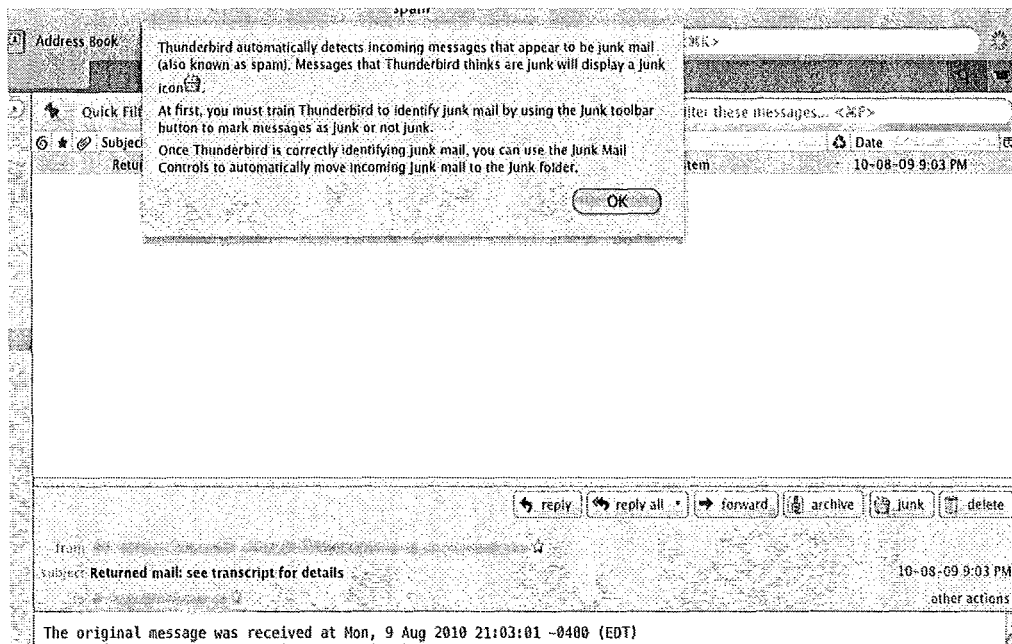


Figure 5 – Screen Capture from Mozilla Thunderbird Spam Guard

2.3 Applications

2.3.1 eHealth

Interest in eHealth, which is the usage of information systems in healthcare, has spread globally and increased greatly in recent years [71]. There are several classes of information systems that address clinical healthcare needs and among them Electronic Medical Records (EMRs) and electronic Personal Health Records (ePHRs) are of most importance to this study. An EMR system is a system that supports healthcare professionals in recording and managing clinical healthcare data electronically [72].

A digital Personal Health Record (ePHR) system is a system that enables patients to access, enter, and manage their own health information online [6]. A good review of different ePHR systems can be found in Halamka

et al [6]. One very interesting ePHR platform is *Indivo*, which is free open source secure software [41]. The system has been deployed in different environments and has evolved gradually into enhanced ePHR systems. Recently Dossia (www.dossia.org) and *Children's Hospital* in Boston have been working together to further improve the *Indivo* platform into a comprehensive ePHR system [6]. The *myOSCAR* system is also an ePHR system which has been developed on the *Indivo* platform by the Department of Family Medicine at McMaster University (see *Figure 6*). *myOSCAR* has several interesting features, ranging from secure messaging to decision support systems [8].



Figure 6 – A Screenshot of the myOSCAR Personal Health Record System

Some of the benefits that can arise from the use of ePHR systems by chronically ill patients include being able to track their diseases in conjunction with their healthcare providers, so that more rapid interventions can be undertaken when problems or deviations occur. Collaborative disease tracking can also improve communication between patients and caregivers, making it easier for patients and caregivers to ask questions, to set up appointments, to request refills and referrals, and to report problems. When an ongoing connection between patient and physician is provided, this

changes encounters from episodic to continuous, and reduces the time needed to address problems that may arise [73].

There are several major challenges in designing and using ePHR systems. These include sharing the appropriate amount of information with patients. In some situations knowing all of the information can be harmful to a patient's health, but studies suggest that there is a positive effect on patient health due to improved doctor-patient communication [5]. Another challenge is supporting secure messaging between patients and their physicians, since patients can easily overwhelm doctors with unnecessary and frequent messages. Another serious challenge is the interoperability or integration of different healthcare systems together, which is a huge problem in the eHealth area. Furthermore, gathering digital information through surveys and trials of ePHR systems can be very helpful for the longterm improvement of healthcare, provided that the uptake of such systems by chronically ill patients is substantially increased. These challenges and many others have been touched upon by Halamka et al [6].

As noted, a challenge that needs to be addressed is the increased rate of communication between patients and healthcare providers. This could increase workloads on already overloaded physicians, resulting in an insuperable barrier to the use of this potentially valuable feature [6]. This is particularly critical in regions where quick access to health professionals is scarce; such as in developing countries [7] and specifically in Africa [74]. This challenge was the main motivation for this work. The proposed system, by semantically triaging patient messages into different priority levels, can help to address this issue.

2.3.2 Opinion Mining

With the exponential growth of Web content in recent years, users from all around the world have the ability to express their views or opinions about almost everything appearing on the Web; ranging all the way from thousands of different products to a multitude of blog posts. Opinion Mining is a related approach to the automated use of mining technology to acquire knowledge from “user-generated content” [4].

Opinion Mining has several interesting applications. One is business intelligence. Businesses are interested in finding out what people are thinking about their products; they are constantly taking surveys to gather user opinions. So mining knowledge about what people are thinking about different products can provide useful business intelligence. This type of

search can result in better designs for products and for Web sites that sell products online. Another application is putting related ads in “user-generated content” [36].

Opinion mining is a customized version of text mining in which the steps are tailored for better performance in mining user opinions. The first customized change to deal with user opinions is at the abstraction level, with specialized features for user opinions instead of normal feature vectors [4, 75]. Because of descriptive words like “great”, “super”, “good”, “bad”, “awful”, etc. in user opinions, a technique has been introduced to distinguish between good and bad reviews, using part-of-speech tagging and semantic orientation [4, 76]. Several other machine learning techniques have been introduced for semantic classification; for example using SVM to distinguish between positive and negative reviews [4, 77]. There have been papers on mining user opinions to find the characteristics of a product that a user was satisfied or not satisfied with, such as Hu and Lio [4, 75, 78]. There have been other published works regarding similar applications such as finding spam or offensive reviews [79].

3. Automated Message Triage (AMT) System

The Automated Message Triage (AMT) system is proposed to semantically triage user messages. The system's goal is to dynamically assign a triage level to a user message, where the triage level will represent its significance for the user. In the healthcare field this significance level can be the urgency of a message; in opinion mining this could be defined as the appeal of a user comment. The triage level can be defined as the application necessitates. After these categories for user inputs have been defined, the system would automatically tailor itself step by step to the users' needs and would be able to improve continuously in selecting the appropriate triage level as more feedback is gathered from users (see *Figure 7*). An interesting feature of the system is that categories can be dynamically changed if required. However, to achieve good results quickly, a set of categorized messages is needed as an initial training base for the application. This initial stage helps AMT to adapt faster to its environment [3].

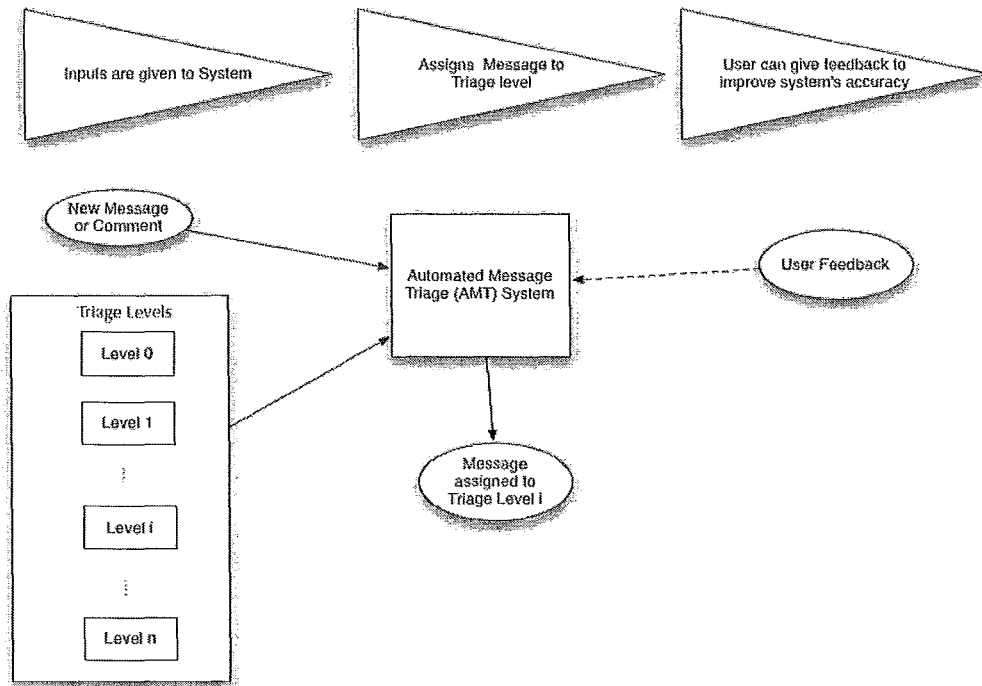


Figure 7 – Automated Message Triage System

The proposed system can be used to triage messages that are coming to a healthcare professional, sort emails and messages that have to be dealt with by a user, sort user comments according to their significance in a blog, choose the best reviews to be shown automatically in a product review page, or even sort news feeds like RSS from the Web according to user interests.

As noted this proposed system can have a variety of applications. Despite the fact that the system has been designed and tested as an application in healthcare field, it can be adapted to other fields requiring this capability, with only minor changes.

3.1 eHealth as the Main Application

As mentioned, the AMT system was inspired by and is designed to cater to the healthcare field. As discussed in 2.3.1, studies have suggested that the use of ePHRs has a moderate and positive effect on patient-doctor-patient communication [5]. *myOSCAR*, an ePHR system, includes a messaging system that allows patients to communicate with their physicians [8] so their physicians can read the messages and take appropriate actions, such as replying to the messages or scheduling appointments to address perceived problems. Because some messages may be low priority or require simple answers, it is important not to overwhelm the physician with these messages [6]. Actually one of the main concerns of healthcare professionals in adopting ePHR systems is the fact that they may be overwhelmed with unwanted and unimportant messages [80]. In a recent trial of *myBP*¹, an adaptation of *myOSCAR* that helped to monitor and manage patients with hypertension, an experienced professional managed the incoming messages from patients through a triage process [81]. This drastically reduced the load on the patient's personal physician by answering the less urgent questions or directing the patient to relevant online information, while forwarding the urgent ones to the attending physician. This is a solution to the physician overload problem, but it can be expensive, since an experienced professional must be on call 24/7 to handle the messages, and as the patient load increases the amount of staff time needed increases proportionately. This makes using an automated system to triage the messages very useful and vital.

One valuable usage for AMT is to automate the triage process so that less urgent messages can be handled automatically, while ensuring that truly urgent messages are forwarded to on-call professionals for rapid response. Because of the sensitive nature of this delicate environment, a high level of

¹ More information about *myBP* trial can be found in 4.1.1

precision is needed to ensure that false negative handling of incoming urgent messages is virtually eliminated. Moreover, this system can also triage text messages from patient cell phones to their health professionals. In this way the connection between patients and their physicians can be even closer with little increase in triage's staff time. This is very important in regions such as developing countries or remote rural areas where access to health professionals is difficult and personnel are scarce [7].

To achieve the goal of the system, a number of messages is needed to train the proposed system so it triage incoming messages effectively. The proposed system will become adapted to its environment as more and more feedback is received from professionals that monitor the system to sort incoming messages. In the next section we will examine AMT system design.

4. Under The Hood

In this section we will go one level deeper into the proposed system and will discuss design and implementation steps of the system. The first step is to review the messages that were used in training and testing the system.

4.1 Data Gathering and Preparation

E-mail messages that were used to test the classifier were gathered during the myBP study. Here is a brief introduction to the myBP trial, how it was done, and why certain methods were used.

4.1.1 myBP Study

Patient Self Management of chronic diseases has an important effect on patient health, both physically and psychologically [82]. According to the Province of Ontario's *Chronic Disease Prevention and Management Framework*, Self Management involves allowing patients to discipline themselves to cope with their diseases [83]. The myBP Study, *Patient Self Management Approach for Hypertension Using Personal Electronic Health Records*, was conducted to determine the effects and the value of using Web-based electronic Personal Health Records (ePHRs) by patients with hypertension [81].

The myBP study was carried out in two phases (see *Figure 8*). The first phase was used to refine the intervention of Web-based ePHRs on patient blood pressure self-management. The second phase was an introductory evaluation of ePHRs for supporting self management [81].

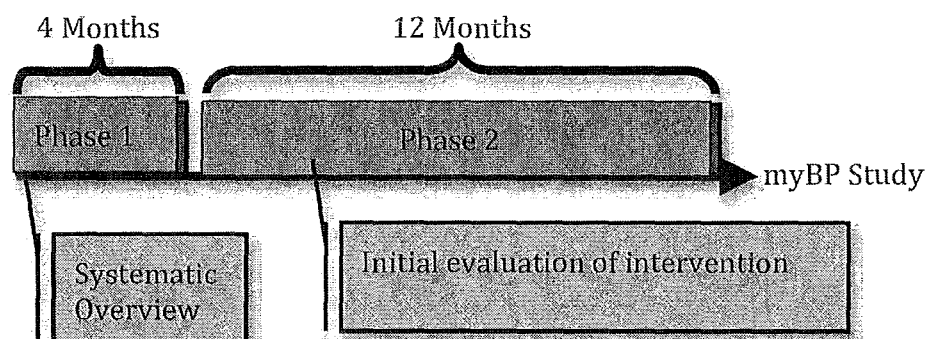


Figure 8 - myBP Study Phases [81]

In the first phase, which took 4 months, an extensive literature review was done and the study was designed as follows. Patients were chosen from a group of people with uncontrolled or undiagnosed high blood pressure [81]. The characteristics of people chosen for the trial appear in Table 4.

Average age	>58 years (average age per group is 58 ± 4 years to approximately 70 years)
Average SBP	Ranged from 120 ± 7 mmHg to 152.2 ± 10.4 mmHg
Average DBP	Ranged from 76 ± 7 mmHg to 89.0 ± 7.9 mmHg

Table 4 - Patient Characteristics Table [81]

Patients were divided into seven trial groups. Five groups were given the same care as before, one group was given some follow-up during the trial using regular mail, and one group (the intervention group) was given access to a Web-based ePHR system. For monitoring their blood pressure, five in the latter group were provided with a blood pressure monitoring device [81].

The intervention group used the myOSCAR platform [81]. myOSCAR is a secure, online health record, based on the Indivo open source, open standard, Web based, and secure [41] personally controlled health record. myOSCAR provides an Application Programming Interface (API). myBP was developed using this API [8]. This system has the potential to become a widely used health self-management platform for patients with chronic diseases. Patients can add/edit or share their health information using myOSCAR, and it can also easily integrate with the OSCAR Electronic Medical Record (EMR) system [8]. OSCAR, (*Open Source Clinical Applications & Resources*), is an open source, open standard, secure EMR that was developed by the Department of Family Medicine at McMaster University and is now being used widely across Canada and internationally [84].

Using the myBP Web-based platform, patients were able to use the features of the myOSCAR ePHR during the study. They could record their blood pressure, see their blood pressure charts to track their progress, and access their healthcare providers (pharmacists, dietitians, nurse practitioners, and primary care physicians) using myOSCAR messaging. Patients took their blood pressure once a week and entered it into myBP. One interesting feature of myBP is its action plans, which are patient pledges to change their lifestyles so they could control their hypertension. Action plans can be designed with myBP software, and myBP gave patients useful

information using the integrated decision support feature to help them to change their lifestyles [81].

4.1.2 Exchanged Messages

As mentioned previously, Web-based application patients can send messages to their health provider. They can directly ask a question or they can report a health issue or a technical problem (see *Figure 9*). A triage person in the myBP trial was a trained clinical assistant or nurse who checked incoming messages and passed them to an appropriate person for action or answers to the messages he or she was capable of responding to [81].

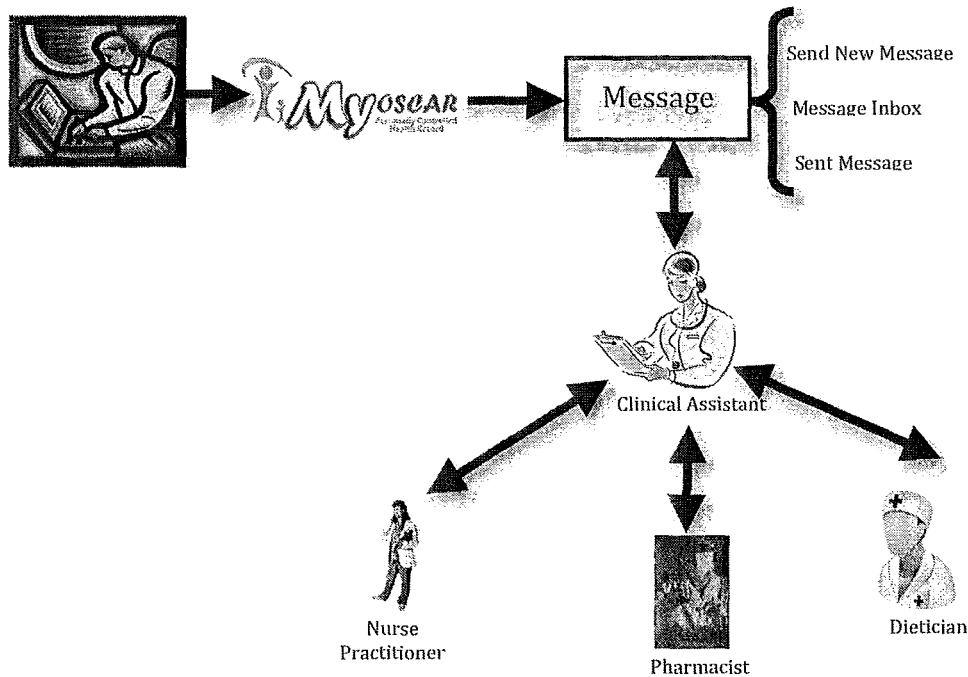


Figure 9 - myBP Messaging System [81]

Other health professionals working with the myBP study checked each patient's health status regularly and took appropriate action. For example if patients were not entering their blood pressure regularly they would be sent a reminder. More details on actions that were taken are listed in Appendix I. During the 12 months of the study about 1460 e-mail messages were exchanged between patients and health professionals. Most were sent

by healthcare professionals during regular checks of patients health status [81].

As the number of patients supported by systems like MyOSCAR increases, the number of messages sent from patients becomes a bottleneck and much more assistant time is needed to support the triage process. To overcome this problem the proposed AMT platform could automatically determine the triage level for incoming messages so health professionals could be automatically informed about urgent messages that needed a response. This way the healthcare staff could focus on important messages and save time otherwise spent responding to non-urgent messages. During the study only a few very urgent messages were actually sent. In a case like this, when health professionals are flooded with messages, the possibility of missing urgent messages gets higher. The AMT platform could help them to focus on truly urgent ones, so the chances of missing urgent messages is reduced, even in the presence of a heavy workload.

The next section focuses on how the information needed for AMT was extracted from the regular message stream.

4.1.3 Data Cleaning

As previously mentioned, about 1460 messages were exchanged between patients and healthcare professionals during the myBP study. The messages were in XML format, but much was information irrelevant to our purposes. A sample XML message can be seen in Appendix II.

The XML file contains information such as sender, receiver, whether the message has been read or not, etc. For training purposes, only the content of the message to be tested and triaged was needed, since message classification is based strictly on message content. A brief description of the steps taken to clean the data follows.

The first and most critical action taken was to protect patient privacy by anonymizing messages and removing private information. Because of the variety of private information and the numerous messages a text mining program called *GATE*[11] was used to detect names, e-mail addresses, and phone numbers in the messages and then a simple Perl script [85] located and replaced them by the XXX symbol, as shown in the sample message in Appendix II. Appendix III demonstrates how *GATE* detects names, and a sample Perl script used to eliminate private information.

To ensure patient privacy after this automatic process a manual review ensured that all the private information had been omitted. At the same time, the messages were flagged with two major triage categories, to distinguish between messages that were worth considering for the study and other messages that were only informative, such as the regularly transmitted messages mentioned previously. This helped the myBP project staff to provide triage classifications for the messages in the study. The cleaned messages were sent to the myBP staff who triaged them as in the following process.

First the messages were divided into different categories. Table 5 contains information about the categories.

Type		Timing	Category
Clinical	Personal	Immediate	A
		Within 24 hours	B
		Within 72 hours	C
	Generic	Within 72 hours	D
Technical (IT) / Data collection		Within 72 hours	E
Duplicate Messages			F

Table 5 - MyBP Patient Message Categories

Based on their priority levels, the messages falling in the different categories were classified into the following triage levels:

Triage Level	Description
Level 0	<i>Immediate action</i> – probable emergency (if recognized by the system, would automatically contact triage person and emergency service). <i>Includes message category A.</i>
Level 1	<i>Immediate action</i> – to be handled by the triage person (if recognized by the system, would automatically contact triage person). <i>Includes message category A.</i>
Level 2	<i>Response within 24 hours</i> – to be handled by triage person (if recognized by the system, or possibly information provided through an automated response). <i>Includes message category B.</i> <u>Examples:</u> Patient appointment times; anything to do with <u>non-urgent</u> elevated / potentially blood pressure readings
Level 3	<i>Response within 72 hours</i> (since the aim is that <u>all messages</u> will be addressed within 3 days). <i>Includes message category C, D and E.</i> <u>Example C:</u> Issues with personal blood pressure monitors <u>Examples D:</u> Medication changes to be updated; updates to other online personal health record <u>Examples E:</u> Issues accessing different aspects / components of online personal health record; difficulties with survey completion; lost passwords; etc
Level 4	<i>Duplicate message</i> – not used for classification

Table 6 - Triage Level Details

Note that:

- If there were two or more issues addressed in a message, the triage level assigned must be for the most urgent issue (ie, the issue associated with shortest response time)
- Triage of messages was based on response time to the message and NOT the time required to provide a solution to the message.

- All messages require a response, even if only an acknowledgement that the message was received or to thank the patient for the information.

Two nurses provided triage levels for 164 of the 1460 messages. The other 1296 messages were mostly automatically generated messages as mentioned in Appendix I and were not retained for this study. To ensure that triage levels assigned to messages were appropriate and that nothing was overlooked, two nurses did the triage in parallel. There were a few differences between the nurse opinions on triage levels that were resolved before using the messages as the training corpus. Appendix IV indicates these differences and how they were addressed.

Another concern was that there were a limited number of patients and the length of the study was relatively short. There were no level 0 messages in the data and only one level 1 message was available, so the training corpus was augmented with simulated messages for these levels. 20 training messages were simulated for each of level 0 and level 1, based on the symptoms and effects of hypertension [86]. These were added to the message corpus and used during the training and testing stages. Appendix IV lists the simulated messages. In the last step all level 0 to 3 messages were saved into separate identified text files for training and testing purposes. Level 4 messages were not used because they were just duplicate messages from other triage levels.

Table 7 shows the number of messages at each triage level that were used for training and testing purposes.

Triage Level	Number of Messages
Level 0	20
Level 1	21
Level 2	19
Level 3	69
Level 4	74

Table 7 - Number of Messages by Triage Level

Figure 10 summarizes the actions taken to prepare the data for training.

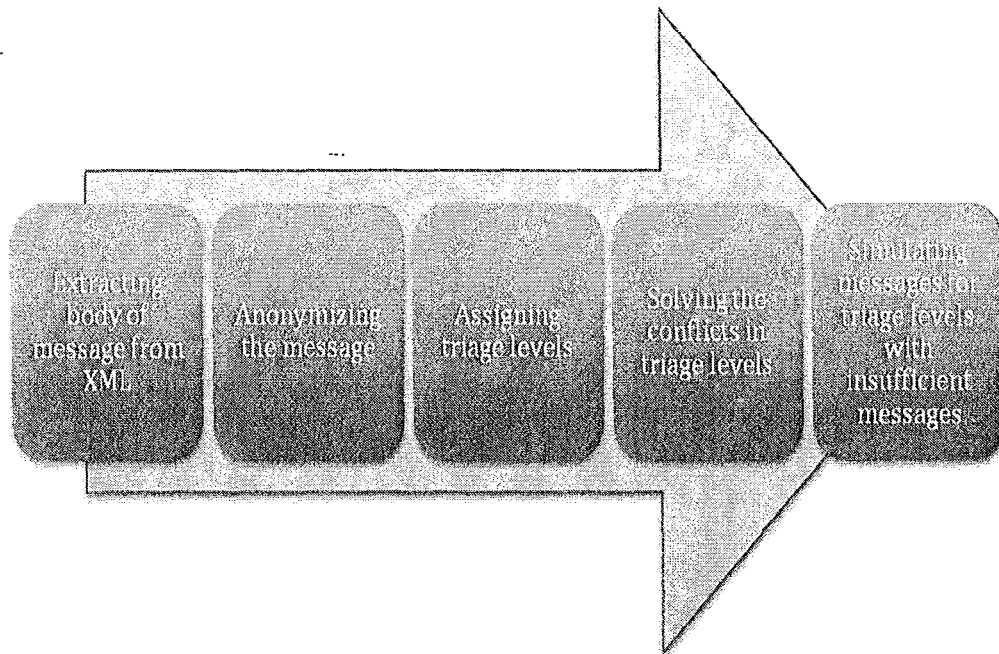


Figure 10 - Steps in Data Preparation

There is no specific method to ensure that the sample size is large enough for a text mining process. The only way to ensure that the sample size is large enough is to check the performance of the system [3]. Given that the results obtained were very satisfactory, it seems that our initial set had an adequate sample size. Details of the proposed system's classification performance are discussed in the next chapter.

4.2. Classification

Predictive modeling [3] is a powerful method of text classification. Text classification (text categorization) consists of two phases. The first phase is to build a model from training documents and the second phase is using the training model to categorize documents. However, features must be extracted from the documents before using them to build or apply the model [1]. The text classification process is shown in *Figure 11*. More information on the variety of feature extraction and classification methods can be found in the literature review in 2.1.1.3 Document Classification.

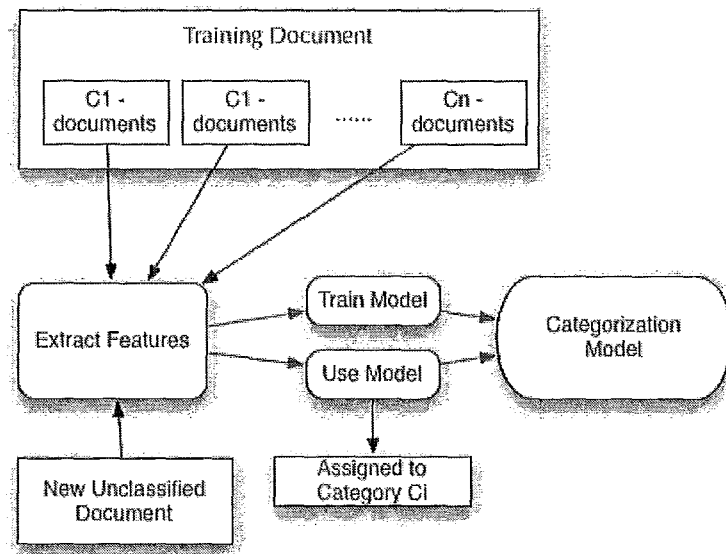


Figure 11 - Text Categorization: Training a Model and Classifying a New Document [1]

As mentioned in 2.1.1.3 Document Classification, numerous linear and non-linear methods have been introduced for text classification. While non-linear methods such as neural networks and support vector machines normally provide better results than linear ones, they are not useful for many practical applications [87], because they lack clarity and completeness. In the proposed system's application domains such as medical or opinion mining, it is very useful to be able to read, evaluate, and optimize the underlying model. Even though this feature has not yet been implemented in our system; it is a useful future extension. Considering this future extension makes linear algorithms like *K Nearest Neighbors* or *Naïve Bayes* the only choices suitable for this work.

For implementing the categorization algorithms there are some well-known open source data mining tools like *GATE* [11] or *WEKA* [88]. Both are Free, Open Source, implemented in Java, and have an Application Programming Interface (API), which allows other programs to use their algorithms and functions as a library. A problem with big packages like *GATE* and *WEKA* is that they are immense applications containing many data mining functions ranging from pattern recognition to knowledge visualization; editing their code to use in desired situations is difficult and challenging. As a result, to implement the proposed system a less extensive Java based library for text mining called *LingPipe* [89] was chosen. *LingPipe* is much smaller than *GATE* or *WEKA*, allowing substantial customizability for

software developers. *LingPipe* provides several libraries for different text mining methods. The code developed for the proposed AMT system using *LingPipe* software is listed in Appendix V.

4.2.1 Classifiers Tested

Among the linear methods like *K Nearest Neighbor* or *Naïve Bayes* that are discussed in chapter 2, linear classification methods needed for the purposes of AMT should address several issues, including learning and converging very fast so that the classifier can be adapted to many environments and that the models are able to be built dynamically enough so categories can be added or removed on the fly. These are important characteristics for AMT's possible applications. For example, healthcare practitioners may change triage levels to fit new unpredicted needs.

The last property of the classification algorithms is that they should be available in an open source library package (i.e. *LingPipe* [89]), so they can be implemented and evaluated in the limited time frame of this work.

Formally, the classification process can be described as:

Suppose $C = (c_1, c_2, \dots, c_m)$ is a set of m categories, and $D = (d_1, d_2, \dots, d_p)$ is a set of p documents. Classification is a function that maps each document in D into a set of categories. So Classification is the function f such that $f: D \rightarrow C$.

Each document is composed of set of words or features². So each document is presented as follows: $d_i = (w_1, w_2, \dots, w_{n_i})$ in which w_1 to w_{n_i} are the set of features extracted from the document.

Table 8 shows the classifiers that were chosen and tested for this work, including their general characteristics. More details about each classifier is provided in separate sections below.

Classifier Type	Data Model	Tokenizer	Feature Extractor
LM Classifier	n-gram based on characters	No Tokenizer	No feature extractor
Naïve Bayes	Bag-of-words	Indo-European	Standard

² See chapter 2 for more information

Classifier		Tokenizer	feature extractor
K Nearest Neighbors	Bag-of-words	Indo-European Tokenizer	Standard feature extractor
TF-IDF Classifier	Bag-of-words	Indo-European Tokenizer	TF-IDF feature extractor

Table 8 – Classifiers Used in This Work

4.2.1.1 Language Model (LM) Classifier

Language Modeling is a very interesting classification method that has been used widely in speech recognition [90]. More recently it has been used in many other areas [91]. One of the simplest and most effective data models for language modeling is the *n-gram* data model [92]. *N-gram* is a sub-sequence of length n of the items given. The *Language Model* rule is to classify a newly given document based on prediction occurring n -grams. If an n -gram has occurred before, the new document is given a higher probability of belonging to that category and, if it has not occurred, less probability is given to that document. The items in *n-gram* can be words or characters. In our case, using *LingPipe*, we built a language model based on character-based *n-gram* [91]. According to the language model, the probability of an item (characters or words) is calculated as [91]:

$$\Pr(w_j | w_{j-n_i+1}, \dots, w_{j-1}) = \frac{|w_{j-n_i+1}, \dots, w_j|}{|w_{j-n_i+1}, \dots, w_{j-1}|}$$

In the formula $|w_{j-n_i+1}, \dots, w_j|$ and $|w_{j-n_i+1}, \dots, w_{j-1}|$ are the number of times the specified n -gram has occurred in the training documents[91].

LingPipe's LM Classifier in the training phase builds a character language model for each category of classification. During the classification phase it calculates the conditional and joint probability of the given item to be classified. In the classification step the joint log probability is calculated as follows [93]:

$$\log_2 \Pr(w_1 \dots w_{n_i}, C) = \log_2 \Pr(w_1 \dots w_{n_i} | C) + \log_2 \Pr(C)$$

Where $w_1 \dots w_{n_i}$ is a character sequence and C is the category. $\Pr(C)$ is calculated using marginal category probability. After calculating this joint probability, the n -gram is scored using cross entropy rates as follows [93]:

$$score(w_1 \dots w_{n_i}, C) = \frac{\log_2 \Pr(w_1 \dots w_{n_i}, C)}{Length(w_1 \dots w_{n_i}) + 2}$$

This score actually is a smoothed version of conditional probability that represents whether the given n-gram belongs to a certain category. Finally, for choosing which category the given n-gram belongs to, we need to find the highest score between categories (i.e. we need to determine $ARGMAX_C \Pr(C, w_1 \dots w_{n_i})$ [93]).

This classifier has been implemented as the *LMClassifier* class in LingPipe. The proposed AMT system uses the *DynamicLMClassifier* class. This is derived from *LMClassifier* and is tailored for active learning applications, satisfying the AMT requirement for a classifier to continue to learn as it is used [93]. The *LM Classifier* has proven to work well in similar applications such as *ePaper* [91] and has also shown good results in the current study. The AMT code that uses this classifier appears in Appendix V.

4.2.1.2 Indo-European Tokenizer

LM Classifier does not use a tokenizer since it uses a character n-gram data model. However, for other classifier “bag-of-words” data models, as described in literature review, are used. In order to extract words (tokens in our case) a simple tokenizer, the *Indo-European Tokenizer*, was used [93]. This tokenizer recognizes the patterns listed in Table 9 as tokens.

Pattern	Description
Numerical	Any sequence of numbers, commas and periods
Hyphen Sequence	Any number of hyphens
Equals Sequence	Any number of equal signs(=)
Double Quotes	Double forward or backward quotes
AlphaNumeric	Any sequence of uppercase or lowercase letters or numbers

Table 9 – Indo-European Tokenizer Tokens [93]

Note that whitespaces are also treated as any sequence of white space characters. It is worth noting here that one great benefit of using this tokenizer instead of an English Tokenizer is the ability to extend the AMT system to use any Indo-European language such as French or Spanish. More details can be found in the *LingPipe* documentation [93].

4.2.1.3 Naïve Bayes Classifier

The *Naïve Bayes Classifier* is a classifier that uses Bayes theorem and assumes a rigorous independence assumption between features. It is commonly used as a classifier in many applications. This classifier is described in Section 2.1.1.3.2 Naïve Bayes Classifier. To use the *Naïve Bayes Classifier* the document must be tokenized in order to extract the feature set. For the proposed AMT application, the Indo-European Tokenizer was used (see above). These tokens are converted into features directly using the *TokenFeatureExtractor* function in *LingPipe* [93]. The reason for not using a feature reduction technique is mentioned in the section Comparison of Classifiers. As noted the *Naïve Bayes Model* assumes independence between tokens (i.e. it uses the “bag-of-words” data model).

In the training phase the feature vectors are saved and when a new document is to be classified the following steps are taken.

According to Bayes rule the conditional probability of a word sequence belonging to a certain category can be determined as follows [93]:

$$\Pr(C | w_1 \dots w_{n_i}) = \frac{\Pr(C) \cdot \Pr(w_1 \dots w_{n_i} | C)}{\Pr(w_1 \dots w_{n_i})} \quad (f1)$$

We are only interested in the numerator since the denominator is constant and can be neglected. $\Pr(C)$ can be calculated using a multivariate estimator [93]. And $\Pr(w_1 \dots w_{n_i} | C)$ is:

$$\Pr(w_1 \dots w_{n_i} | C) = \prod_{i=1}^{n_i} P(w_i | C)$$

To calculate $P(w_i | C)$ *LingPipe* uses a formula based on the *Dynamic Language Model*, (see the Language Model Classifier section), and smooths the token model for better prediction results [93]. Now as mentioned in formula (f1) to the following is calculated in order to determine the probability that a certain feature vector belongs to a specific category. So the result $ARGMAX_C \Pr(C) \cdot \Pr(w_1 \dots w_{n_i} | C)$ gives the category that the given sequence of words belongs to [93]. Figure 12 from [1] shows the entire process as described:

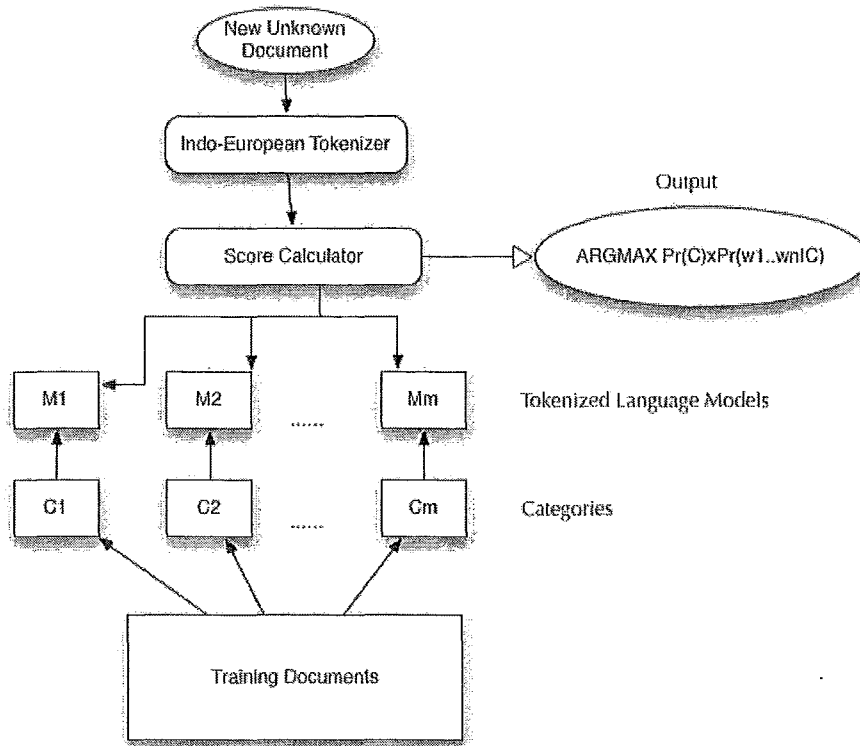


Figure 12 - Classification Process of Naive Bayes Classifier [1]

This classification is implemented as the *NaiveBayesClassifier* class in *LingPipe*, a direct derivative of the *DynamicLMClassifier* mentioned in the Language Model Classifier section. A more detailed description of this function appears in the *LingPipe* documentation [93]. The code used in AMT to implement this method is in Appendix V.

4.2.1.4 K Nearest Neighbor (Knn) Classifier

The *K Nearest Neighbor* (Knn) Classifier uses the *K Nearest Neighbor* algorithm to classify messages. A detailed description of this classifier is in Section 2.1.1.3.1 *K Nearest Neighbor*. A customized version of the Knn algorithm that was used in this work, is as follows.

In the training phase the algorithm stores feature vectors of the training examples along with their categories. The features are extracted with the “bag of words” model, using the *Indo-European Tokenizer* [93]. This phase is the same used for the *Naive Bayes Classifier* (see the *Naive Bayes Classifier* section).

In order to classify a new document (see *Figure 13*), the distances between the new document's feature vector and the vectors saved in the training phase are calculated. In the current case, after trying different distance functions, the normal Euclidean distance algorithm was chosen due to its better performance. Details of the performance measures calculated are in the Comparison of Classifiers section. The Knn classifier for calculating distance between vectors uses a concept called proximity [93], calculated as follows:

$$\text{proximity}(d_i, d_j) = \frac{1}{1 + \text{distance}(d_i, d_j)}$$

This proximity measure scales the distance function to a positive real number in [0..1] to avoid dividing by zero. The *LingPipe* distance function always returns absolute positive values, and calculates proximity between the new document and all of the training documents. Then it chooses the first k documents with the highest proximity to the new document's feature vector. Finally it assigns the new document to the category that appears most frequently among its k neighbors [93].

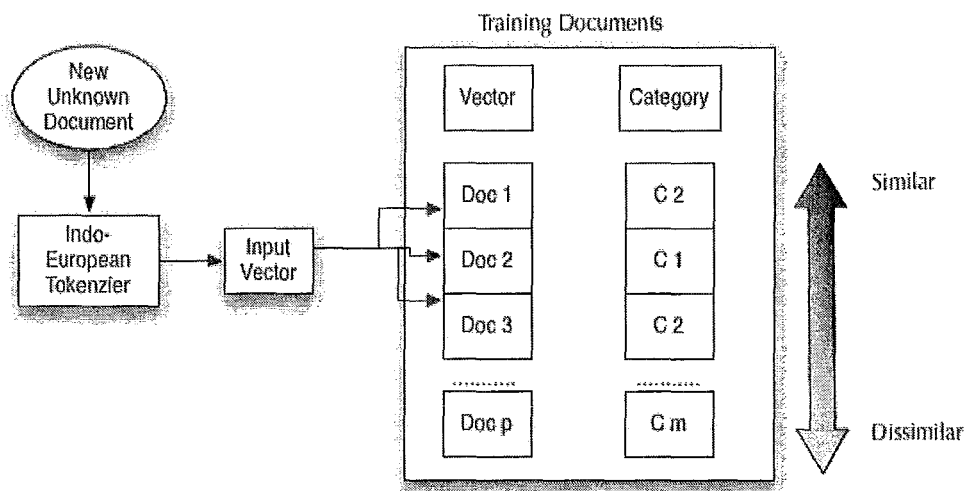


Figure 13 - Knn Classifier [1]

The Knn Classifier is known to work well in very irregular environments [93], and this describes the nature of the environment we are targeting for the AMT system. This classifier has been implemented in *LingPipe* using *KnnClassifier* class. A more detailed description can be found

in *LingPipe* documentation [93]. Its implementation code for AMT appears in Appendix V.

4.2.1.5 TF-IDF Classifier

The *TF-IDF Classifier* is based on calculations of Term Frequency (TF) and Inverse of Document Frequency (IDF) to classify messages. More detailed information and similar versions of TF-IDF are in Section 2.1.1.2 Feature Reduction or Selection. TF-IDF is mostly known as a feature reduction technique; however, in *LingPipe* it has been tweaked to fit as a classifier.

LingPipe's TF-IDF classifier training phase is similar to that used for the Knn and Naïve Bayes classifiers. First, features are extracted from the training documents using the Indo-European Tokenizer. These feature vectors are smoothed using the TF-IDF measure, which is based on token frequency and its inverse document frequency [1]. This normalization is different from feature reduction; in feature reduction the undesired features are eliminated, but here the undesired features are not omitted and may only be set aside when not needed.

The training phase using TF-IDF distance is as follows.

If w_j is a feature in category, c_p , then the *inverse document frequency* is [93]:

$$idf(w_j) = \ln\left(\frac{df(w_j)}{m}\right)$$

Here, $df(w_j)$ is the *document frequency* of feature w_j , or the number of categories in which feature w_j has occurred, and m is the total number of categories. Thus, the more a feature is used in different categories the less weight will be given to it. *term frequency* is defined as [93]:

$$tf(d_j, w_i) = \sqrt{\text{count}(d_j, w_i)}$$

In which $\text{count}(d_j, w_i)$ is count of w_i feature in d_j document. Square roots are used to normalize the *term frequency* factor. Then TF-IDF weights are saved into a weight vector \vec{v} , calculated as [93]:

$$\vec{v}[d_j][w_i] = tf(d_j, w_i) \times idf(w_i)$$

This vector is saved and used as a reference in the classification phase.

In the classification phase, the TF-IDF distance measure is used to classify a new document. The first step involves extracting tokens using the token extractor as mentioned before, and the features are saved in a feature vector called \vec{f} . TF-IDF weights are calculated in the same way and saved in vector \vec{x} [93].

$$\vec{x}[f_i] = tf(f_i, current_doc) \times idf(f_i)$$

Afterwards the vector cosine distance score is calculated for each category [93]:

$$score(\vec{v}[d_j], \vec{x}) = \cos(\vec{v}[d_j], \vec{x}) = \frac{\vec{v}[d_j] \cdot \vec{x}}{|\vec{v}[d_j]| \cdot |\vec{x}|}$$

Here, the numerator is a normal dot product of the two vectors, and the denominator is a number that represents the product of the two vector sizes. The cosine of the two vectors is 0 if they are orthogonal, 1 when they point in the same direction, and -1 when they are pointing in opposite directions. Therefore the result will be a number between 1 and -1, representing the new vector's fitness to be assigned to the candidate category. The category with the highest score is used to classify the new document [93]. The TF-IDF classification process is shown in *Figure 14*.

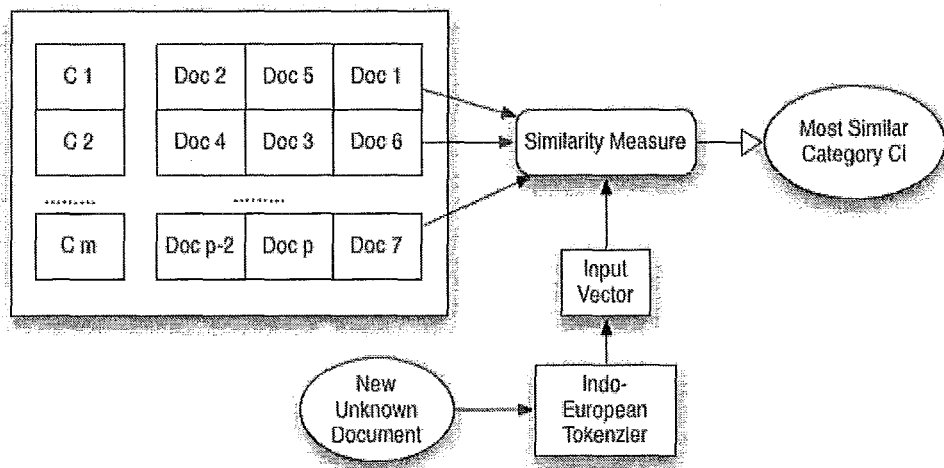


Figure 14 – TF-IDF Classifier Classification Process [1]

TF-IDF classification is implemented in *LingPipe* as *TfidfClassifierTrainer* class. More detailed implementation notes are in *LingPipe*'s documentation [93]. The AMT system's code that implements this classifier is in Appendix V.

4.2.2 Combining Classifiers

After trying single classifiers to classify the documents, gaining greater performance by combining classifiers is a proven approach. Section 2.1.1.4 Multiple Classification Combination, touches on combining multiple classifiers and similar systems. Three bagging methods for combining classifiers were chosen for this work and one of the combination methods was tailored for this work's environment in order to gain even greater performance. The tailored new method proved to perform better than any other combination methods, through rigorous tests on our training corpus. Details of each of these combination methods are described in their specific sections below, and in the Comparison of Classifiers section they are compared not only among each other but also with single classifiers. Finally, the best classification method was chosen to be the basis of the proposed AMT system.

The combination methods used were a bag of classifiers already mentioned, including:

1. Language Model Classifier
2. Naïve Bayes Classifier
3. K Nearest Neighbor Classifier
4. TF-IDF Classifier

The combination methods used included:

1. Simple Voting
2. Dynamic Classifier Selection
3. Adaptive Classifier Combination
4. New Adaptive Classifier Combination

4.2.2.1 Training Multiple Classifiers

Training phases for the classifiers used were the same as training each separate classifier, as already discussed. The training algorithm basically involved training the separate algorithms one by one. After the training phase all the training information was available for each of the individual classifiers.

4.2.2.2 Simple Voting

As mentioned in 2.1.1.4 Multiple Classification Combination, simple voting is just classifying the new document to the category that most classifiers have voted for [26] (the process is shown in *Figure 15*). However, in the AMT system the simple voting algorithm was tweaked to avoid situations where votes are equal, thus adding better performance to this simple method.

In our tweaked version of the simple voting algorithm a score is calculated for each category through each classifier's weighted vote in order to avoid ties in voting. These weighted votes are based on the classifier's assurance when selecting that specific category, calculated as the probability of the classifier's choice for its best category. The main benefit of this method is that it avoids confusion and breaks ties in voting, which can occur when votes for each category are the same (e.g., when all classifiers choose different categories). In these cases choosing the category for which the classifier is more certain is a good approach.

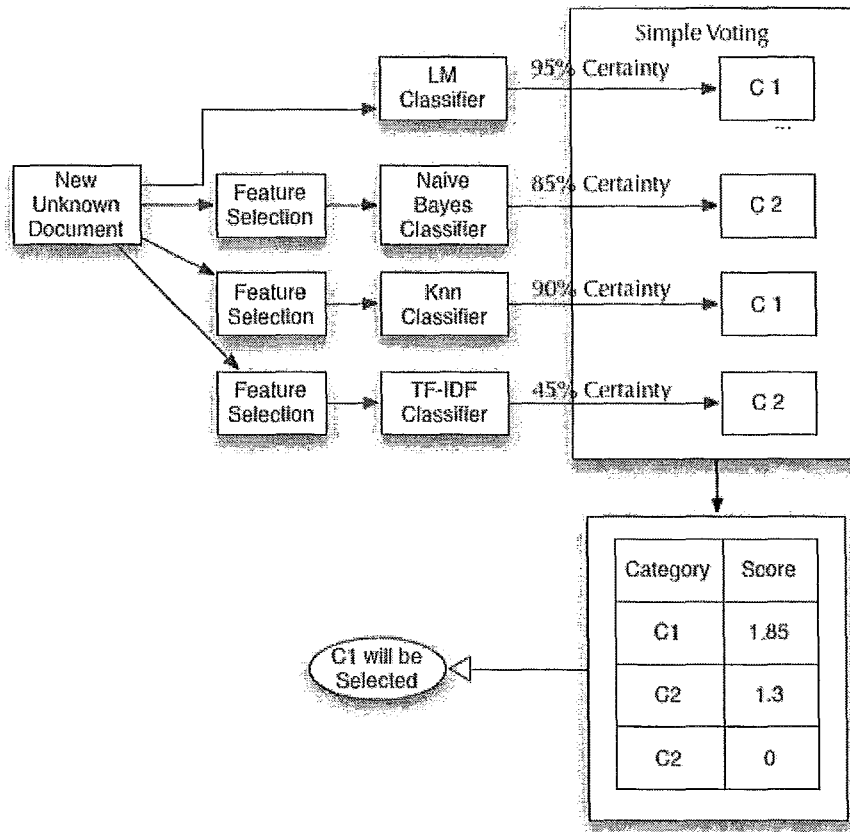


Figure 15 - Simple Voting Classifier Combination with Modification

Assurance levels are calculated differently for different classifiers. With *Naïve Bayes* and *LM Classifier* the probability calculated can be used directly because these numbers represent the certainty of the classifier when it chooses a certain category. This probability is also known as posteriori probability [58]. On the contrary for *Knn* and *TF-IDF* classifiers, the measure of comparison is a score which cannot be used directly to represent assurance level. For these algorithms, the scores for the different categories were summed and the current score was divided by the corresponding sum to simulate the probability representation as follows.

$$\Pr(c_i) = \frac{score(c_i)}{\sum_j score(c_j)}$$

Details of code implementation for this classifier are in Appendix V.

4.2.2.3 Dynamic Classifier Selection

Dynamic Classifier Selection is based on the concept of choosing the classifier that has been best at classifying a new document [58]. See *Figure 2* to see a good demonstration of Classifier Selection. More discussion about this method is in 2.1.1.4 Multiple Classification Combination. Dynamic Classifier Selection in AMT is based on work by Li et al [26]. The classifier with the best local precision is chosen, based on its performance at k nearest neighbor selection of the new document.

To implement this algorithm, the k nearest neighbors of the new document are chosen in the training data, and then the classifier that has best precision in the neighborhood is selected to classify documents. Both normal Euclidean distance and the so-called soft [26] cosine distance were used in this work to calculate the neighborhood of the new document, with no significant differences between the two results. More details about dynamic classifier selection appear in the literature review. Code for this classifier is given in Appendix V.

4.2.2.4 Adaptive Classifier Combination

The demonstration of classifier combination is available in *Figure 3*. This application in AMT was inspired by the discussion on Adaptive Classifier Selection by Li [26]. As mentioned in 2.1.1.4 Multiple Classification Combination, classifier combination is a method of combining results from different classifiers to get better performance.

The pseudocode for adaptive classifier combination is as follows [26]:

1 – Find the k nearest neighborhood for the new document from the training data. In the AMT system $NB(new_doc) = (nb_1, \dots, nb_k)$ is calculated using the Euclidean distance k nearest neighbor method.

2 – The new document is classified using all of the n classifiers. The categories selected are $C' = (c'_1, c'_2, \dots, c'_n) \in (c_1, c_2, \dots, c_m)$

3 – For all categories $c'_i \in C'$ we calculate the following:

$$Acc_i = \sum_{s=1}^n \sum_{j=1}^k \cos(nb_j, new_doc) \times \Pr(c'_i | nb_j \in c'_i)$$

where $\cos(nb_j, new_doc)$ is the cosine distance between vectors obtained from the n classifiers and the newly given document and $\Pr(c'_i | nb_j \in c'_i)$ is the

probability that nb_j neighborhood documents have been assigned to c'_j . The method for calculating this *posteriori* probability is described in the section on simple voting combination.

4 – Finally the new document is classified to category c'_u where $u = ARGMAX_i(Acc_i)$

The implementation code for this multiple classification method is in Appendix V.

4.2.2.5 New Proposed Combination Method

The performance of the adaptive combination method has been improved with the following adjustment. Since finding neighbors which are closest to the given vector has proven to be an NP-Hard problem according to the Closest Vector Problem (CVP) [94], we aimed at simply improving the neighborhood locating method as much as possible. This revised classifier is similar to the Adaptive Classifier Combination and the first step of this classifier is tweaked where the neighborhood is being located. If the Euclidean distance method is substituted for the Cosine distance method no significant change occurs in the classifier's performance. Not even changing the method to other more radical methods like Taxicab distance [38] will change its performance. After reviewing the training corpus it was found that a given document's neighborhood does not have a great similarity to the given document in most cases because first, there are a relatively small number of messages in some categories and second, the nature of our environment which is basically sentences from patients. To overcome this problem we used the soft cosine measure for finding the neighborhood with cap of .70 on similarity. This is the cosine of 45 degrees, which means that documents that are at least half similar to each other will be selected as the neighborhood. However, using this cap is likely to reduce the precision of the classifiers being used because most of the documents will not be assigned any neighborhood and will be out of range of this cap. Softening the cap will not improve the classifier's precision and might make it difficult to choose an appropriate neighborhood. Therefore an innovative method was introduced to solve the problem. In the new approach the 45 degree similarity cap is used, but if there are fewer than two neighbors for the new document, the two best neighbors are chosen as the document's neighborhood. This adjustment proved to be very effective and the Adaptive Classifier Selection

with the neighborhood-locating tweak showed better results than the other methods on almost all the test cases run.

The pseudocode for the neighborhood-locating method is:

1 – Use the cosine measure to locate the k nearest neighbors of the new document.

2 – When there are more than two documents in the neighborhood set do the following step, starting with the least similar neighborhood document.

3 – Check to see if the similarity of the new document to the neighborhood document is less than 0.7. If it is, then omit that neighborhood from the set that will be considered.

The result of this method is compared to other methods in the next section. The code used for this method is in Appendix V.

4.2.3 Comparison of Classification Methods

This section compares different classifiers, using measures that were introduced in 2.1.1.5 Classifier Performance. A new type of error is also introduced, which is the "critical error". This type of error occurs when a level 0 message (an absolute emergency) is assigned a lower priority level category (i.e. a higher numerical level) or if a level 1 message is put into a lower level category. Other cases, such as putting a non-emergency message into an emergency level or misplacing level 2 and 3 messages instead of each other are not as critical in the healthcare environment, so critical error (Cerr) is a good measure for comparing classifiers in this application. Cerr is calculated as:

$$C_{err} = \frac{\text{number of level0 messages in other levels} + \text{number of level1 messages in level2 or 3}}{\text{count of level0 messages} + \text{count of level1 messages}}$$

It is also worth noting here that recall measure, as described in 2.1.1.5 Classifier Performance, is measured for each class separately. The recall measure, which is used for classifier comparisons, is average recall of all categories, a good representation of recall for all categories, and calculated as follows [93].

$$\frac{1}{m} \sum_{i=1}^m \text{recall}(c_i)$$

in which m is number of categories and $recall(c_i)$ is recall for a specific category.

To begin the comparison, benchmark conditions are discussed, as follows.

4.2.3.1 Benchmark Conditions

In order to compare different classification methods and their precision, the set of training messages was used. 80% of the messages were chosen randomly as training instances and the other 20% for testing the system. The current time in milliseconds was used as a randomization seed to make sure that the choices were not biased toward a specific randomization seed. Then each of the algorithms was run 100 times to get a good average and eliminate any effects of the randomization process. After testing the use of differing numbers of total runs by incrementing the number of runs gradually, the results did not show any significant change as the number of runs increased beyond 100.

The number of messages in each triage level is given in *Table 7*. Obviously, these numbers vary according to triage level. This issue may bias classifiers towards certain categories, an undesirable situation; nonetheless this possibility is inevitable in almost all real world training data [1].

Code for these tests can be found in the *main* function of the code in Appendix V. The sample output of the program that was used to generate these results is available in Appendix VI. All of the measures used to compare classifiers and in the following discussion are based on 100 run tests.

4.2.3.2 Classifier Confusion Matrices and Test Parameters

A classifier confusion matrix is a simple matrix representing how accurately the classification was done, and the parameters that the tests ran. Among the different test conditions, the different classifiers were compared using several measures.

4.2.3.2.1 Language Model (LM) Classifier

For this algorithm one parameter, the size of the n-gram, needs to be set. As discussed, this algorithm uses a character based n-gram to classify messages so an appropriate size should be the average length of a word. After some tests with different numbers (see *Figure 16*), 7 was chosen as the n-gram size for the LM Classifier. This is also a good average for word length.

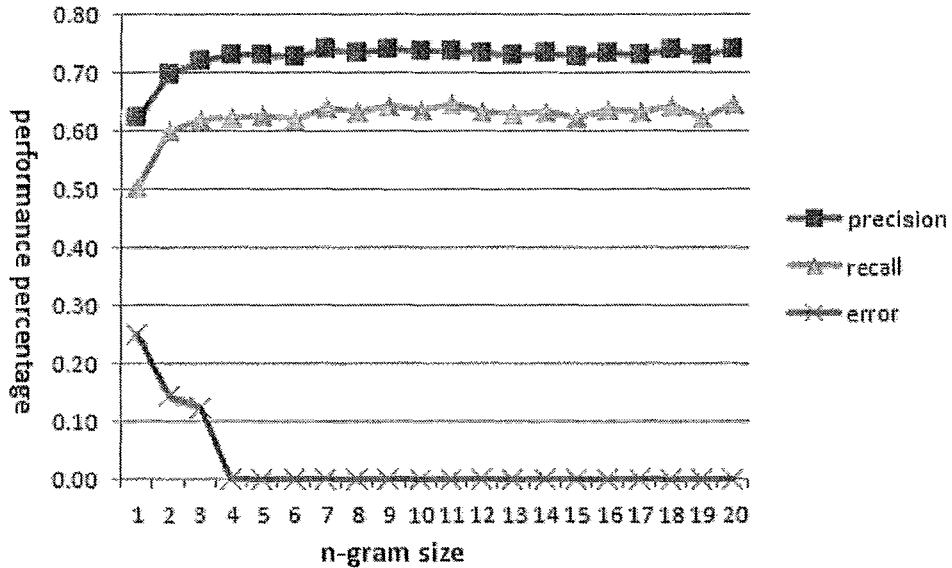


Figure 16 - n-gram Size for LM Classifier Performance

The chart indicates that increasing the n-gram size beyond 7 will not result in significant changes in classifier precision. This classifier had the least error rate and best precision among all the classifiers used, so it was the best single classifier of choice for this study. Table 10 shows the Confusion Matrix for the results from this classifier in 100 runs, using an n-gram size of 7.

		Response			
		Level 0	Level 1	Level 2	Level 3
Reference	Level 0	3	0	0	0
	Level 1	1	3	0	0
	Level 2	0	0	0	4
	Level 3	0	0	0	14

Table 10 - Language Model (LM) Confusion Matrix (n-gram size = 7)

4.2.3.2.2 Naïve Bayes Classifier

This algorithm has no parameter to set. On average it demonstrated reasonable performance. Table 11 shows the Confusion Matrix for this classifier...

		Response			
		Level 0	Level 1	Level 2	Level 3
Reference	Level 0	3	0	0	0
	Level 1	1	3	0	0
	Level 2	0	0	0	4
	Level 3	0	0	0	13

Table 11 - Naïve Bayes Confusion Matrix

4.2.3.2.3 K Nearest Neighbor (Knn) Classifier

In this classifier there are two parameters to set; one is the method of finding the neighborhood (i.e. to measure the distance between the new document and training corpus documents used) and the other is the number k chosen for the number of neighbors. Figure 17 shows the effect of different k on classifier performance.

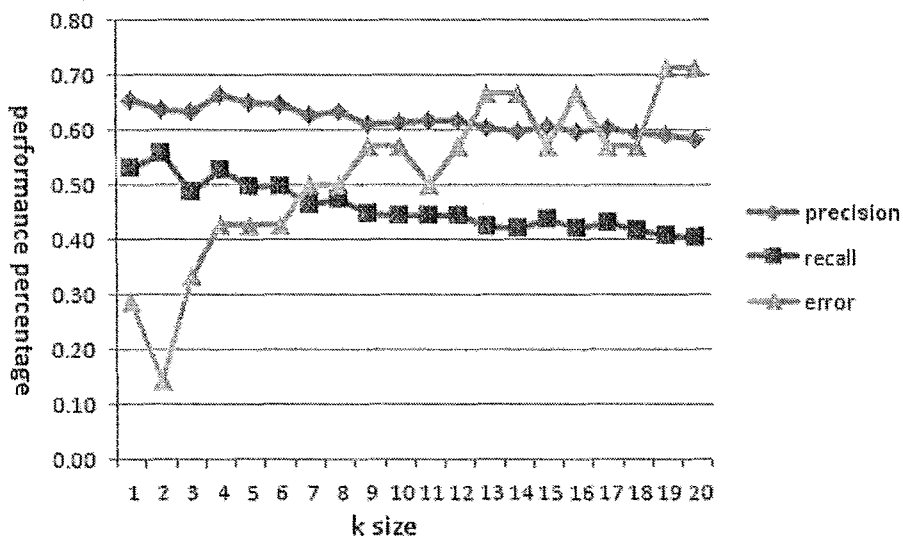


Figure 17 – k Size to Knn Classifier Performance

Figure 17 shows that, on balance, the best number to choose for k is 2, since C_{err} is at a minimum and precision and recall are reasonable. The reason for this behavior is described partially on 4.2.2.5 *New Proposed Combination Method*. The fact that our training corpus is small and documents similar to the new document are more difficult to find easily, suggests that it is better to choose a small number like 2 in this case.

After testing different distance functions like Taxicab and Minkowski with different orders, none was superior to the classic Euclidean method. Table 12 shows the results for different neighborhood methods with 100 runs for $k=4$. This k value was chosen to make the effect of neighborhood distance function clear and measurable, while at the same time with $k =4$ precision and recall were still reasonable.

Distance Function	Precision	Recall	Cerr
Euclidean	.66	.53	.43
Taxicab	.59	.39	.85
Minkowski order 3	.64	.51	.33
Minkowski order 4	.63	.51	.29
Minkowski order 5	0.61	.50	.29

Table 12 - Effects of DistanceFunction on Knn Classifier

The reason for this behavior might be due to the small size of the training corpus. It is difficult to find good neighbors to new documents in this situation. Table 13 shows the Confusion Matrix with $k = 2$ and with the Euclidean distance function used to find the neighborhood.

		Response			
		Level 0	Level 1	Level 2	Level 3
Reference	Level 0	3	0	0	0
	Level 1	1	2	0	1
	Level 2	0	0	1	3
	Level 3	1	0	1	12

Table 13 - Knn Confusion Matrix ($k=2$, Euclidean Distance)

4.2.3.2.4 TF-IDF Classifier

The TF-IDF classifier has no parameter to set; however it is worth noting here the reason for not using any feature reduction algorithm in this work. After some tests it became clear that using any feature reduction method, including TF-IDF or stop word reduction, can cause significant damage to the system's precision. For instance using the TF-IDF filter will drop LM classifier precision with the same n-gram size from 74% to about 50% precision. The reason is that text documents in this case are relatively short messages which often contain only insignificant features, and there is no perfect feature reduction method that will not result in losing valuable features. A good algorithm to deal with this situation is the current TF-IDF algorithm. This algorithm does not omit any features so, if at some point a particular feature is needed it can be used; the features are just weighted based on their TF-IDF measures during classification. This is the main reason for this classifier's good performance in this environment. Table 14 shows the Confusion Matrix for this method after 100 runs.

		Response			
		Level 0	Level 1	Level 2	Level 3
Reference	Level 0	3	1	0	0
	Level 1	0	3	0	0
	Level 2	0	0	0	3
	Level 3	0	0	1	12

Table 14 – TF-IDF Confusion Matrix

4.2.3.2.5 Simple Voting Classifier

There is no reason to believe that the Simple Voting Classifier would work well in any environment including this work; however it is a good comparison point for classifier combinations. Table 15 shows the Confusion Matrix for this method after 100 runs.

		Response			
		Level 0	Level 1	Level 2	Level 3
Reference	Level 0	4	0	0	0
	Level 1	1	3	0	0
	Level 2	0	0	0	4
	Level 3	0	0	0	13

Table 15 – Simple Voting Classifier Confusion Matrix

4.2.3.2.6 Dynamic Classifier Selection

As for the *Knn Classifier*, we need to select two parameters for the *Dynamic Classifier*. The first is the number of neighbors *k* that are chosen for the local decision set. Results are shown in *Figure 18*.

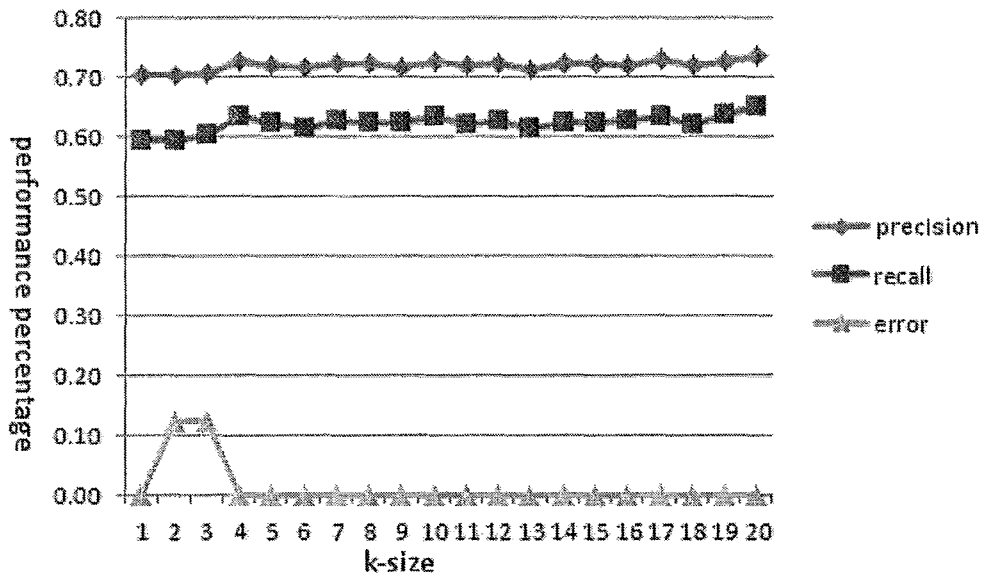


Figure 18 - *k* Dynamic Classifier Selection (DCS) Classifier Performance

Figure 18 shows that setting *k* = 4 will reduce the chance of a *Cerr* problem, and there is a peak in chart precision and recall for this value. Note that there were 100 runs for each value of *k* in the chart, so the results should change very little for new document category selection. The second parameter can be selected by trying different neighborhood methods to

calculate the local neighborhood with $k=4$. Table 16 shows the results from these tests.

Distance Function	Precision	Recall	Cerr
Euclidean	.73	.63	0.00
Cosine	.71	.61	0.00
Taxicab	.72	.61	0.00
Minkowski order 3	.72	.61	0.00

Table 16 – Different Distance Functions for Calculating DCS Neighborhoods (with $k=4$)

It can be concluded from Table 16 that the best distance function to use with this method is Euclidean distance with $k=4$. There was significant error resistance from this method in different situations, but it demonstrated no performance gain over the best single classifier methods which were the LM Classifier or the Adaptive Classifier Selection methods that we tested. Table 17 shows the Confusion Matrix for DCS, using $k=4$ and Euclidean distance.

		Response			
		Level 0	Level 1	Level 2	Level 3
Reference	Level 0	3	0	0	0
	Level 1	0	3	0	0
	Level 2	0	0	0	4
	Level 3	0	0	0	13

Table 17 – Dynamic Classifier Selection (DCS) Confusion Matrix ($k=4$ and Euclidean Distance)

4.2.3.2.7 Adaptive Classifier Combination

As for Dynamic Classifier Selection, in the Adaptive Classifier Combination there are two parameters to select; k and distance function. Figure 19 shows the effect of changing k on this classifier's performance.

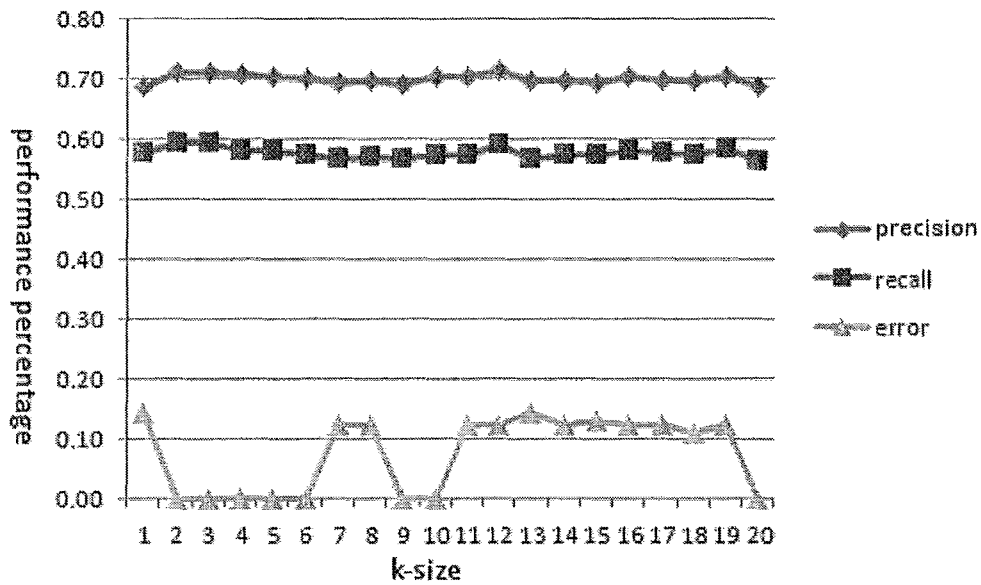


Figure 19 - k Size to Adaptive Classifier Combination Classifier (ACC) Performance

Figure 19 demonstrates that a value of $k=4$ is a suitable choice, since its precision and recall is good, and the error is 0.00. Keeping the error rate down is essential to avoid critical errors as much as possible in the operating environments being considered for AMT. $k=4$ was set to choose the best neighborhood methods, as demonstrated in Table 18.

Distance	Precision	Recall	Cerr
Euclidean	.71	.58	0.00
Cosine	.73	.65	0.00
Taxicab	.67	.53	.14
Minkowski order 3	.7	.58	0.00

Table 18 - Different Distance Functions for Calculating ACC Neighborhoods ($k=4$)

Table 18 indicates that the soft cosine distance measure seems to have a positive effect on this classifier's performance. This effect might be due to the unpredictable nature of the environment where a "good" neighborhood is difficult to find for a new document. This seems to make soft measures more powerful in choosing better neighborhoods. However, this slight increase in performance can be overcome using the new method, to be discussed shortly. Table 19 shows the Confusion Matrix for the adaptive classifier selection method, with $k=4$ and using the cosine distance measure.

		Response			
		Level 0	Level 1	Level 2	Level 3
Reference	Level 0	4	0	0	0
	Level 1	1	3	0	0
	Level 2	0	0	0	4
	Level 3	0	0	0	13

Table 19 – Adaptive Classifier Selection (ACC) Confusion Matrix ($k=4$, Cosine Distance Measure)

4.2.3.2.8 New Adaptive Classifier Combination

In the new adaptive classifier combination it is only necessary to choose the k parameter, since the soft cosine measure has already been chosen. An algorithm based on this measure was developed. *Figure 20* demonstrates the performance of this classifier for different values of k .

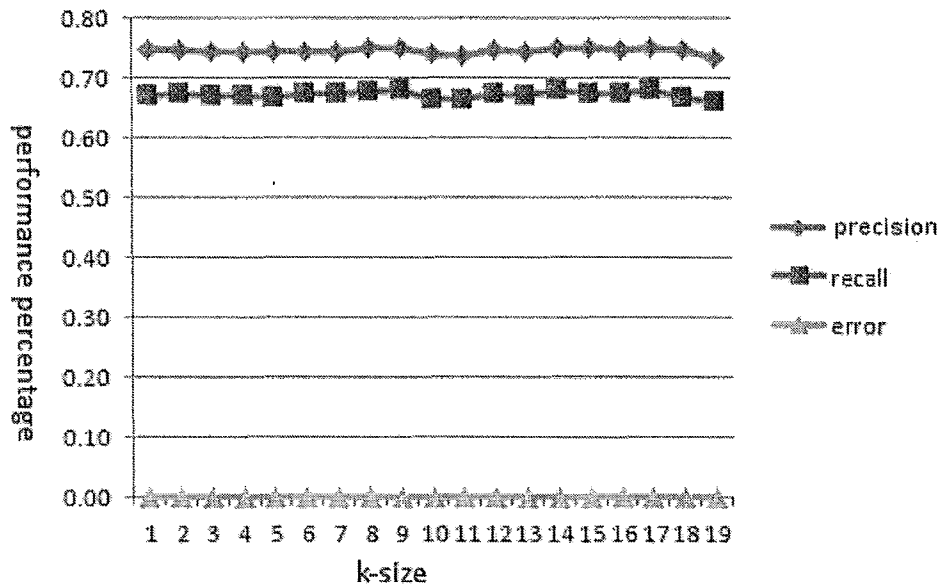


Figure 20 - k Size to New Adaptive Classifier Combination Classifier Performance

Figure 20 can be interpreted as showing that this classifier not only offers better performance than other methods, single or multiple, it is also very resistant to making critical errors. The reason is that the Dynamic Classifier Selection method always chooses one classifier to classify messages

so it is less prone to making critical errors. The Adaptive Classifier Combination method is also very strong in combining classifier results to get high precision, but it is prone to making bad judgments in some situations. Giacinto offers a good comparison of these two methods [56].

The problem with the Adaptive Classifier Combination in this case might be seen as the “bad neighborhood”. When these so called bad neighborhoods, which are not even 50% similar to a new document, are in the decision set, they can drive the algorithm to make critical mistakes. By enhancing the algorithm to find neighborhoods, we can benefit from the performance and precision of the Adaptive Classifier Combination and avoid these errors. Error avoidance is absolutely essential to this work's applications; by just allowing desirable “good” neighborhoods into the final decision set, this enhances the likelihood of the correct choice. This change makes this method better than either of the normal adaptive classifier selection and dynamic classifier selection methods for the type of application considered in this research. It is worth noting here that this method is a better approximation for finding neighbors of the newly given document although it is not mathematically the perfect solution for finding neighbors or combining the classifiers. This method uses the cosine distance measure that assumes the lengths of most messages are not substantially different, as applied to our current training corpus and most other similar applications. If length of the messages and consequently the derived vectors are substantially different, this should also be taken into consideration. The Confusion Matrix for this algorithm is shown in Table 20 with $k=9$. $k=9$ was chosen because of a slight performance enhancement that the algorithm demonstrated for this value.

		Response			
		Level 0	Level 1	Level 2	Level 3
Reference	Level 0	4	0	0	0
	Level 1	1	3	0	0
	Level 2	0	0	0	3
	Level 3	0	0	0	13

Table 20 – New Adaptive Classifier Combination Confusion Matrix ($k=9$)

4.2.3.3 Choosing the best method

After representing the Confusion Matrix for each method separately Table 21 compares the different classifiers at their best.

Algorithm	Precision	Recall	Cerr	Parameters
Language Model Classifier	.739	.638	.000	n-gram = 7
Naïve Bayes Classifier	.690	.582	.000	N/A
K Nearest Neighbor Classifier	.643	.563	.143	K=2 using Euclidean distance
TF-IDF Classifier	.706	.645	.143	N/A
Combination Methods³				
Simple Voting Classifier	.720	.621	.000	N/A
Dynamic Classifier Selection Classifier	.726	.624	.000	K=4 using Euclidean distance
Adaptive Classifier Selection	.733	.648	.000	K=4 using Cosine similarity measure
New Adaptive Classifier Selection	.749	.671	.000	K=9

Table 21 – Classifier Comparison

Table 21 shows further evidence of precision for the new algorithm in finding appropriate neighborhoods for Adaptive Classifier Selection in this environment.

³ In the version of Knn Classifier that we used in multiple classifiers, we chose k as 4 and not 2. When the Knn Classifier is combined with other classifiers its precision is a great help to the overall system's precision, and the other classifiers can correct possible critical errors. Selecting k as 4 is the better choice since choosing k as 2 has a negative effect on multiple classifier performance. We selected 7 as the LM Classifier's n-gram size in multiple classifiers, which is similar to its single classifier n-gram size.

4.3 Connecting the Dots

After finding a good way to classify messages that suit the environment, the remaining task is to put the appropriate components together and build the final system. *Figure 21* is the overall design of the proposed system.

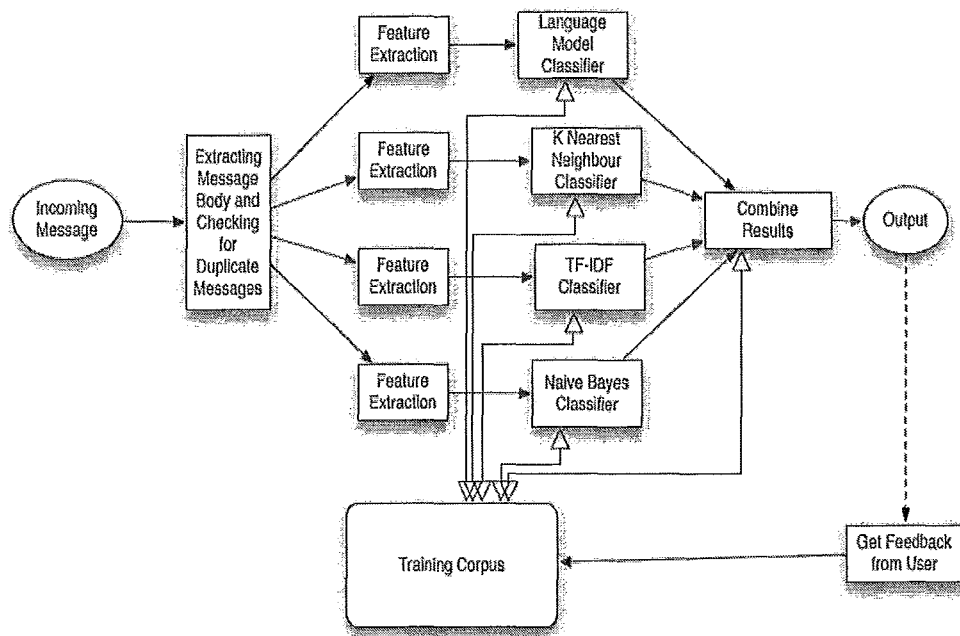


Figure 21 - Automated Message Triage Overall Design

At the centre of the AMT system is the modified Adaptive Combination Method that uses the new neighborhood algorithm as the classifier of choice. The system can be trained on an initial message set to become adapted to the new environment, and it will improve over time as more feedback is received from users. It is very dynamic. A classifier can be omitted or be replaced by another classifier according to the environment's need without a significant change in the system. If users decide to change categories in the system they would just need to reset the system and train it with new categories, and the system would be ready to classify the new categories. All in all, AMT is a fast, dynamic, accurate and error resistant system for semantically sorting text documents.

5. Future Work

There are several directions for moving ahead and achieving better performance. One proven method is to add a classifier to the list of our classifiers that is known to work well in the environment where AMT would be operating. If suitable to the expected environment, this new classifier could use techniques that have been used in opinion mining, like sentimental classification (being aware of sentimental words that are used in classification) [4]. Adding these classifiers could only improve performance since combining classifiers ensures that the system uses the results from the best classifier when other classifiers fail to perform well. It is also useful to use a classifier that not only classifies based on the words mostly used in that specific environment; it adaptively finds out what those words are, using techniques discussed by Sanchez-Graillet et al [52].

Because the classifier training corpuses are human readable and we are using linear methods, one useful extension to AMT would be a program to read the classifiers' underlying model and let users change manually how the classifiers are trained. This option can be very useful in specialized fields like healthcare where healthcare professionals can easily find whether a relationship between two words is the correct one. This task can be done automatically even if the system has a good knowledge of the nature of that environment. For example, in healthcare if the program is able to understand systems like Systematized Nomenclature of Medicine-Clinical Terms (SNOMED-CT) [95] then it is able to use such systems to improve classifier performance without any user feedback.

Triaging in the myBP trial included about 44% generic messages. If the system could go one step farther to handle generic messages automatically, it could become even more useful in areas with healthcare professional shortages [74] or it could handle online education of patients about their chronic diseases automatically. The AMT system can be also be generalized to handle text messages from patient cell phones. This capability is absolutely essential in rural areas and remote communities since it can bring instant emergency responses to patients in need of help, and save the time and cost of healthcare professionals.

Our research results could also be used in the Opinion Mining field. The first step would be to categorize a training set of reviews based on their importance; the resulting system could then determine the importance of new reviews and user opinions. The system could be customized to automatically find desirable reviews, neglecting whether they are positive or

negative. This could then be applied to find only positive reviews or negative ones. It would be useful to automatically select important reviews to show on a Web product page or to sort comments mentioned on a blog post or a product page. If user reviews needed to be triaged this system could save much time in sorting them according to their significance.

Another interesting way to extend the system would be using the AMT system to prioritize emails that are sent to the user. Google recently released Priority Inbox [96], which is a system to sort emails based on their importance for user. This shows how interesting and important this feature is for current web users. AMT System can be trained based on priority levels that are defined by user needs. As the user gives more and more feedback the system can perform better at prioritizing the emails. Consequently, the AMT system could be tailored readily to accomplish this interesting and useful feature.

BIBLIOGRAPHY

1. Konchady, M., *Building Search Applications, Lucene, LingPipe, and Gate*. 2008, Oakton, Virginia: Mustru Publishing. 430.
2. Han, J. and M. Kamber, *Data mining : concepts and techniques*. Second ed. Morgan Kaufmann series in data management systems. 2006, San Francisco: Morgan Kaufmann Publishers. xxiv, 550 p.
3. Weiss, S., Indurkha, N., Zhang, T., Damerou, F., *Text Mining: Predictive Methods for Analyzing Unstructured Information*. 2004: Springer, Heidelberg
4. Liu, B., *Opinion Mining*, in *Web Data Mining*. December, 2006, Springer. p. 532.
5. Ross, S.E. and C.T. Lin, *The effects of promoting patient access to medical records: a review*. J Am Med Inform Assoc, 2003. **10**(2): p. 129-38.
6. Halamka, J.D., K.D. Mandl, and P.C. Tang, *Early experiences with personal health records*. Journal of the American Medical Informatics Association : JAMIA, 2008. **15**(1): p. 1-7.
7. Hall JJ, T.R., *Health for all beyond 2000: the demise of the Alma Ata Declaration and the primary health care in developing countries*. Medical Journal of Australia, 2003. **178**: p. 17-20.
8. *myOSCAR Manual*. 2010; Available from: <http://myoscar.org/users-manual-1>.
9. U. Fayyad, G.P.-S., P. Smyth., *From Data Mining to Knowledge Discovery in Databases*. Ai Magazine. **17**(3): p. 37-54.
10. Feldman, R. and J. Sanger, *The text mining handbook : advanced approaches in analyzing unstructured data*. 2007, Cambridge ; New York: Cambridge University Press. xii, 410 p.
11. H. Cunningham, K.B., V. Tablan, D. Maynard, H. Saggion, W. Peters, N. Aswani, I. Roberts, A. Funk, D. Damjanovic, M. Agatonovic, T. Heitz, A. Roberts, M. Greenwood, S. Szasz, G. Gorrell. *GATE Website - General Architecture fore Text Engineering*. 2010; Available from: www.gate.ac.uk.
12. Apte, C., et al. *Towards Language Independent Automated Learning of Text Categorization Models*. in *In Proceedings of the 17th Annual ACM/SIGIR conference*. 1994.
13. Balabanovic, M. and Y. Shoham. *Learning Information Retrieval Agents: Experiments with Automated Web Browsing*. 1995; 13-18]. Available from: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.2509>.

14. Bartell, B.T., G.W. Cottrell, and R.K. Belew, *Latent semantic indexing is an optimal special case of multidimensional scaling*, in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*. 1992, ACM: Copenhagen, Denmark. p. 161-167.
15. Berry, M.W., S.T. Dumais, and G.W. O'Brien, *Using linear algebra for intelligent information retrieval*. *SIAM Rev.*, 1995. **37**(4): p. 573-595.
16. Joachims, T., *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization*, in *Proceedings of the Fourteenth International Conference on Machine Learning*. 1997, Morgan Kaufmann Publishers Inc. p. 143-151.
17. Yang, Y., *Expert network: effective and efficient learning from human decisions in text categorization and retrieval*, in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. 1994, Springer-Verlag New York, Inc.: Dublin, Ireland. p. 13-22.
18. Mladenic, D. *Personal WebWatcher: design and implementation*. 1996; Available from:
<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.2143>.
19. Armstrong, R., et al. *WebWatcher: A Learning Apprentice for the World Wide Web*. in *AAAI Spring Symposium on Information Gathering*. 1995.
20. Cohen, W.W., *Learning to Classify English Text with {ILP} Methods*, in *Advances in Inductive {L}ogic {P}rogramming*, L. De Raedt, Editor. 1996, IOS Press. p. 124-143.
21. Lewis, D. and W. Gale. *A sequential algorithm for training text classifiers*. in *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. 1994. Dublin, Ireland: Springer-Verlag New York, Inc.
22. Pazzani, M., J. Muramatsu, and D. Billsus. *Syskill & Webert: Identifying interesting web sites*. in *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*. 1996.
23. Sorensen, H., M. McElligott, and M. Elligott. *PSUN: A Profiling System for Usenet News (Extended Abstract)*. Available from:
<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.3209>.
24. Aizawa, A., *An information-theoretic perspective of tf-idf measures*. *Information Processing & Management*, 2003. **39**(1): p. 45-65.
25. Salton, G. and C. Buckley, *Term Weighting Approaches in Automatic Text Retrieval*. 1987, Cornell University.
26. Y.H. Li, A.K.J., *Classification of Text Documents*, in *Proceedings of the 14th International Conference on Pattern Recognition-Volume 2 - Volume 2*. 1998, IEEE Computer Society. p. 1295.

27. Yiming, Y. and O.P. Jan, *A Comparative Study on Feature Selection in Text Categorization*, in *Proceedings of the Fourteenth International Conference on Machine Learning*. 1997, Morgan Kaufmann Publishers Inc.
28. Salton, G. and M.J. McGill, *Introduction to Modern Information Retrieval*. 1986: McGraw-Hill, Inc. 400.
29. Sch, H., et al., *A comparison of classifiers and document representations for the routing problem*, in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. 1995, ACM: Seattle, Washington, United States. p. 229-237.
30. Lutu, P.E.N. and A.P. Engelbrecht, *A decision rule-based method for feature selection in predictive data mining*. *Expert Syst. Appl.*, 2010. **37(1)**: p. 602-609.
31. J. A. E. Weston, A.E., B. Schoelkopf, F. Perez-Cruz, *Methods for feature selection in a learning machine* G. Health Discovery Corporation (Savannah, Editor. 2009: United States.
32. Jain, A. and D. Zongker, *Feature selection: evaluation, application, and small sample performance*. *Ieee Transactions on Pattern Analysis and Machine Intelligence*, 1997. **19(2)**: p. 153-158.
33. Baker, L.D. and M. Andrew Kachites, *Distributional clustering of words for text classification*, in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 1998, ACM: Melbourne, Australia.
34. N. Slonim, N.T., *The Power of Word Clusters for Text Classification*. 23rd European Colloquium on Information Retrieval Research, 2001.
35. David, D.L., *An evaluation of phrasal and clustered representations on a text categorization task*, in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*. 1992, ACM: Copenhagen, Denmark.
36. David, D.L., *Representation and learning in information retrieval*. 1992, University of Massachusetts: Amherst.
37. Gang, Q., et al., *Similarity between Euclidean and cosine angle distance for nearest neighbor queries*, in *Proceedings of the 2004 ACM symposium on Applied computing*. 2004, ACM: Nicosia, Cyprus.
38. Krause, E.F., *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*. New edition ed. 1987, Menlo Park, California: Dover Publications Inc. 88.
39. Pang-Ning, T., S. Michael, and K. Vipin, *Introduction to Data Mining, (First Edition)*. 2005: Addison-Wesley Longman Publishing Co., Inc.
40. Charu, C.A., *Towards systematic design of distance functions for data mining applications*, in *Proceedings of the ninth ACM SIGKDD*

- international conference on Knowledge discovery and data mining.* 2003, ACM: Washington, D.C.
41. Mandl, K.D., et al., *Indivo: a personally controlled health record for health information exchange and communication.* BMC Medical Informatics and Decision Making, 2007. **7**: p. -.
 42. Pedro, D. and P. Michael, *On the Optimality of the Simple Bayesian Classifier under Zero-One Loss.* Mach. Learn., 1997. **29**(2-3): p. 103-130.
 43. Yiming, Y. and G.C. Christopher, *An example-based mapping method for text categorization and retrieval.* ACM Trans. Inf. Syst., 1994. **12**(3): p. 252-277.
 44. Genkin, et al., *Large-Scale Bayesian Logistic Regression for Text Categorization.* Technometrics, 2007. **49**(3): p. 291-304.
 45. Hinrich, S., et al., *A comparison of classifiers and document representations for the routing problem,* in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval.* 1995, ACM: Seattle, Washington, United States.
 46. Wiener, E., J. Pedersen, and A. Weigend. *A neural network approach to topic spotting.* in *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval.* 1995.
 47. Miguel, E.R. and S. Padmini, *Hierarchical neural networks for text categorization (poster abstract),* in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval.* 1999, ACM: Berkeley, California, United States.
 48. Huang, C.-L. and C.-J. Wang, *A GA-based feature selection and parameters optimization for support vector machines.* Expert Systems with Applications, 2006. **31**(2): p. 231-240.
 49. Tai-Yue, W. and C. Huei-Min, *Fuzzy support vector machine for multi-class text categorization.* Inf. Process. Manage., 2007. **43**(4): p. 914-929.
 50. Timo Koski, J.M.N., *Bayesian Networks - An Introduction.* Wiley series in probability and statistics. 2009: John Wiley & Sons, Ltd.
 51. Mieczys, A.K. aw, and opotek, *Very large Bayesian multinets for text classification.* Future Gener. Comput. Syst., 2005. **21**(7): p. 1068-1082.
 52. O. Sanchez-Graillet, M.P., *Acquiring Bayesian Networks from Text,* in *Proceedings of LREC.* May 2004: Lisbon.
 53. Sandeep, R. *Bridging Text Mining and Bayesian Networks.* 2009.
 54. Cohen, W. *Text Categorization and Relational Learning.* in *In Proceedings of the Twelfth International Conference on Machine Learning.* 1995.

55. Cohen, W. and Y. Singer. *Context-Sensitive Learning Methods for Text Categorization*. in *Acm Transactions on Information Systems*. 1996.
56. Giacinto, G. and F. Roli, *Adaptive selection of image classifiers*, in *Image Analysis and Processing*, A. Del Bimbo, Editor. 1997, Springer Berlin / Heidelberg. p. 38-45.
57. Hansen, L.K. and P. Salamon, *Neural Network Ensembles*. IEEE Trans. Pattern Anal. Mach. Intell., 1990. **12**(10): p. 993-1001.
58. Giacinto, G. and F. Roli, *Dynamic Classifier Selection*, in *Multiple Classifier Systems*. 2000, Springer Berlin / Heidelberg. p. 177-189.
59. Woods, K., K. Bowyer, and W.P. Kegelmeyer, Jr. *Combination of multiple classifiers using local accuracy estimates*. in *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*. 1996.
60. Jianpei, Z., C. Lili, and M. Jun, *A New Multiple Classifiers Combination Algorithm*, in *Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences - Volume 2 (IMSCCS'06) - Volume 02*. 2006, IEEE Computer Society.
61. Leah, S.L. and W.B. Croft, *Combining classifiers in text categorization*, in *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. 1996, ACM: Zurich, Switzerland.
62. Giacinto, G. and F. Roli, *Dynamic Classifier Selection based on Multiple Classifier Behaviour*. Pattern Recognition, 2001. **34**: p. 1879-1881.
63. Huang, Y.S. and C.Y. Suen, *A Method of Combining Multiple Experts for the Recognition of Unconstrained Handwritten Numerals*. IEEE Trans. Pattern Anal. Mach. Intell., 1995. **17**(1): p. 90-94.
64. Saerens, M. and F. Fous, *Yet Another Method for Combining Classifiers Outputs: A Maximum Entropy Approach*, in *Multiple Classifier Systems*, F. Roli, J. Kittler, and T. Windeatt, Editors. 2004, Springer Berlin / Heidelberg. p. 82-91.
65. Yang, Y., *An Evaluation of Statistical Approaches to Text Categorization*. Information Retrieval, 1999. **1**(1): p. 69-90.
66. McKeown, K., et al. *Tracking and Summarizing News on a Daily Basis with Columbia's Newsblaster*. 2002; Available from: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.9542>.
67. Sable, C. and K. Church. *Using Bins to Empirically Estimate Term Weights for Text Categorization*. in *In Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01)*. 2001.
68. *Collaboration, Electronic messaging, Anti-Abuse and Spam Conference [formerly the Conference on Email and Anti-Spam] CEAS*. 2010; Available from: <http://ceas.cc>.

69. *Thunderbird*. 2010; Available from: <http://www.mozillamessaging.com/en-US/thunderbird/>.
70. Graham, P. *Better bayesian filering*. in *Proceedings of the 2003 Spam Conference*. 2003.
71. Gerber, T., et al., *An agenda for action on global e-health*. Health affairs, 2010. **29**(2): p. 233-6.
72. Kristiina, H.y., S. Kaija, and N.n. Pirkko, *Definition, structure, content, use and impacts of electronic health records: A review of the research literature*. International Journal of Medical Informatics, 2008. **77**(5): p. 291-304.
73. Tang, P.C., et al., *Personal Health Records: Definitions, Benefits, and Strategies for Overcoming Barriers to Adoption*. Journal of the American Medical Informatics Association, 2006. **13**(2): p. 121-126.
74. Ross, S.J., D. Polsky, and J. Sochalski, *Nursing shortages and international nurse migration*. Int Nurs Rev, 2005. **52**(4): p. 253-62.
75. Mingqing Hu, B.L., *Mining opinion features in customer reviews*, in *Proceedings of the 19th national conference on Artificial intelligence*. 2004, AAAI Press: San Jose, California.
76. Peter, D.T., *Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews*, in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. 2002, Association for Computational Linguistics: Philadelphia, Pennsylvania.
77. Kushal, D., L. Steve, and M.P. David, *Mining the peanut gallery: opinion extraction and semantic classification of product reviews*, in *Proceedings of the 12th international conference on World Wide Web*. 2003, ACM: Budapest, Hungary.
78. M. Hu, B.L. *Opinion Feature Extraction Using Class Sequential Rules*. in *In Proc. of the Spring Symposia on Computational Approaches to Analyzing Weblogs (AAAI-CAAW-06)*. 2006.
79. Ee-Peng Lim, V.-A.N., Nitin Jindal, Bing Liu and Hady Lauw, *Detecting Product Review Spammers using Rating Behaviors*, in *The 19th ACM International Conference on Information and Knowledge Management (CIMK-2010, full paper)*. 2010: Toronto, Canada.
80. Norm Archer, U.F.-T., *An Empirical Study of Canadian Consumer and Physician Perceptions of Electronic Personal Health Records*, in *ASAC 2010 Conference*. 2010: Regina, Sask.
81. *Patient Self Management Approach for Hypertension Using Personal Electronic Health Records (myBP)* unpublished paper. 2010, Department of Family Medicine at McMaster University: Hamilton, Ontario.
82. Bodenheimer, T., et al., *Patient self-management of chronic disease in primary care*. JAMA, 2002. **288**(19): p. 2469-75.

83. Care, M.o.H.a.L.-T. *Chronic Disease Prevention and Management*. May 2007; Available from: <http://www.health.gov.on.ca/english/providers/program/cdpm/index.html>.
84. *OSCAR Canada Users Society Website*. 2010; Available from: <http://www.oscarcanada.org/>.
85. *The Perl Programming Language*. Available from: <http://www.perl.org/>.
86. *Symptoms of Hypertention*. Available from: <http://www.symptoms-of-hypertension.com/>.
87. Baesens Bart, K.U.L., Martens David, Setiono Rudy, Zurada Jacek,, *Special issue on white box nonlinear prediction models*. IEEE Transactions on Neural Networks, 2010. **21**(3): p. 527-527.
88. Ian H. Witten, E.F., *Introduction to Weka*, in *Data Mining: Practical Machine Learning Tools and Techniques*. 2005.
89. Carpenter, B. *LingPipe – a suite of Java libraries for the linguistic analysis of human language*. Available from: <http://alias-i.com/lingpipe/>.
90. *Language Modeling for Speech Recognition - Microsoft Research*. Available from: <http://research.microsoft.com/en-us/projects/language-modeling/default.aspx>.
91. Lena Tenenboim, B.S., Peretz Shoval, *Ontology-Based Classification Of News In An Electronic Newspaper*, in *International Conference "Intelligent Information and Engineering Systems" INFOS 2008*: Varna, Bulgaria.
92. Peng, F., D. Schuurmans, and S. Wang, *Language and task independent text categorization with simple language models*, in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. 2003, Association for Computational Linguistics: Edmonton, Canada. p. 110-117.
93. *LingPipe 4.0 API Documentation (JavaDoc)*. 2010; Available from: <http://alias-i.com/lingpipe/docs/api/index.html?overview-summary.html>.
94. Micciancio, D., *Closest Vector Problem*, in *Encyclopedia of Cryptography and Security*, H. Tilborg, Editor. 2005, Springer US. p. 79-80.
95. *SNOMED CT (Systematized Nomenclature of Medicine-Clinical Terms)* Available from: <http://www.ihtsdo.org/snomed-ct/>.
96. *Gmail Priority Inbox - Get through your email faster*. Available from: <http://mail.google.com/mail/help/priority-inbox.html>.

Appendices

Appendix I

Actions taken in myBP study on different situations:

- Perform daily checks using the "Report: BP entries" function *Survey Administration* in MyOSCAR to determine which patients have not entered any BP values in the past 7 days.
- For patients who have entered BP values, review the values they have entered by viewing their MyOSCAR record

- Patients will be sent a message based on their BP reading once per week **unless** their BP reading is greater than 179/109 mmHg*; the message will be based on their **highest** reading of the week
 - o If their highest reading is <140/90 mmHg (or <130/80 mmHg if patient has diabetes), a Congratulatory message is to be sent:
"Great job on continuing to monitor your blood pressure! Your blood pressure reading on [date] was [reading], which is below your blood pressure target. Congratulations!"

 - o If their highest reading is between 141/91 mmHg and 179/109 mmHg, please send them this message:
"Congratulations on continuing to monitor your blood pressure!

Your blood pressure reading on [date] was [reading], please check your blood pressure again in the next few days. If your blood pressure is still in this range, you should make an appointment to see your family physician within the next week."

- *If their daily blood pressure reading was between than 180/110 to 209/119 mmHg, send a message to them **the same day**. The message should read:
"Your blood pressure reading on [date] was [reading], you should make an appointment to see your family physician within the next few days. I will follow-up with you to ensure you have made this appointment. If you are concerned about your blood pressure reading, urgent study support can be reached at 416-464-3995"

- *If their daily blood pressure reading was higher than 210/120 mmHg, send a message to them **the same day**. The message should read:

“Your blood pressure reading on [date] was [reading], you should arrange a same day appointment with your family physician or go to the emergency room. If you are concerned about your blood pressure reading, urgent study support can be reached at 416-464-3995”

Reminder Messages for Entry of BP Readings

1. If patients have not entered a blood pressure reading within the past **7 days**, send a Reminder message (*BP Monitoring Reminder Message*)
2. If the first reminder sent to a patient does not lead to the patient entering a BP reading within **3 days**, a second reminder email will be sent to the patient. (*BP Monitoring Reminder Message*)
3. If the patients still has not entered a BP reading another **3 days**, a third reminder email will be sent. (*BP Monitoring Reminder Message*)
4. Reminder emails will be sent every 7 days thereafter until the patient enters a BP reading. (*BP Monitoring Reminder Message*)

Appendix II

A sample of the XML in the raw myBP study messages

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:IndivoDocument
xmlns:ns2="http://indivo.org/xml/phr/document">
<ns2:DocumentHeader>

<Author>
  <IndivoId>project.support@myoscar.org</IndivoId>
  <Name>XXX</Name>
  <Role>provider</Role>
</Author>

<ns2:DocumentClassification>
  <Classification>urn:org:indivo:document:classification:messa
ge</Classification>
</ns2:DocumentClassification>

<ns2:ContentDescription>
<ContentType
xmlns:ns3="http://indivo.org/xml/phr/message">ns3:Message</Conte
ntType>
</ns2:ContentDescription>

  <Active>true</Active>
</ns2:DocumentHeader>

<ns2:DocumentVersion>
<ns2:VersionHeader>
  <VersionAuthor>
    <IndivoId>project.support@myoscar.org</IndivoId>
    <Name>XXX</Name>
    <Role>provider</Role>
  </VersionAuthor>
</ns2:VersionHeader>

<ns2:VersionBody>
<ns2:Message xmlns:ns2="http://indivo.org/xml/phr/message">
  <Recipient> 1-2-24@myoscar.org</Recipient>
  <Subject>Cardiovascular risk factor survey </Subject>
  <ContentType>ns2:TextMessage</ContentType>
  <Read>>false</Read>
  <Replied>>false</Replied>
  <MessageContent>
  <ns2:TextMessage>
  Dear XXX,
```


Creating an action plan is great way to get the process started to control your blood pressure! Please complete the Blood Pressure Risk Assessment Survey so you can identify which lifestyle factors you should to target for your personal action plan.

We also ask that you complete Medication Use Survey that is in the Survey list.

Thank you,

```
Project Support
</ns2:TextMessage>
</MessageContent>
</ns2:Message>
</ns2:VersionBody>

</ns2:DocumentVersion>
</ns2:IndivoDocument>
```

Please note that the nn's that appear everywhere are actually carriage returns (enters).

Appendix III

For detecting names, addresses and other private information, GATE (*General Architecture for Text Engineering*) [11], is a very useful free, open source, user friendly, text mining tool. Using this tool is described fully in GATE's website [11]. This is a short introduction about how we used GATE was used to find private information in messages.

The opening GATE screen looks like this.

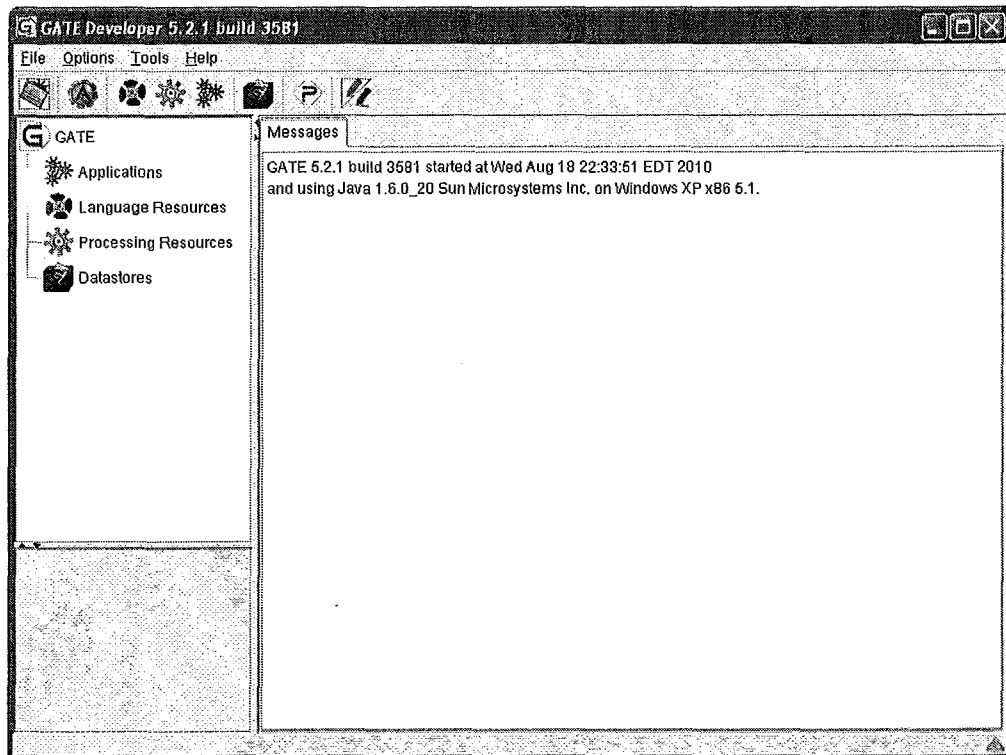


Figure 22 – GATE Opening Screen

The text documents are first added as a corpus to the program, by right clicking Language Resources, selecting New Document and adding a document to GATE. A new corpus is built by drag-and-dropping that document in the corpus, until all documents are added to the corpus.

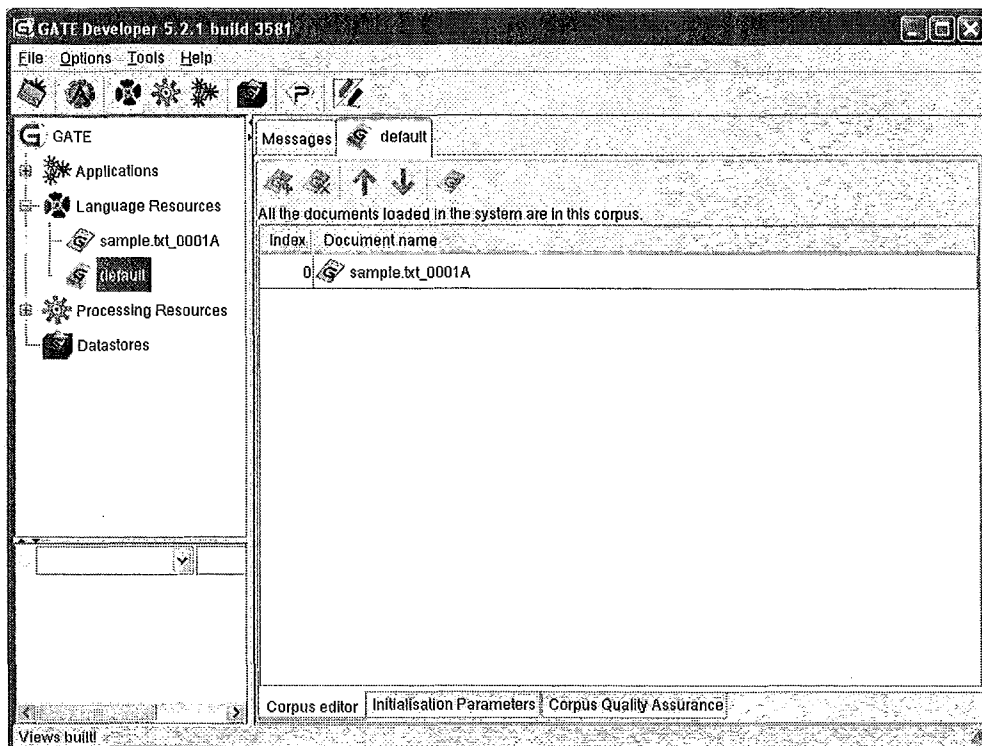


Figure 23 – GATE with Document Corpus

Then the *ANNIE* application was added to the program by selecting it from File->Load *ANNIE System...* ->with defaults. After doing so the *ANNIE* application was selected and ran on the current corpus.

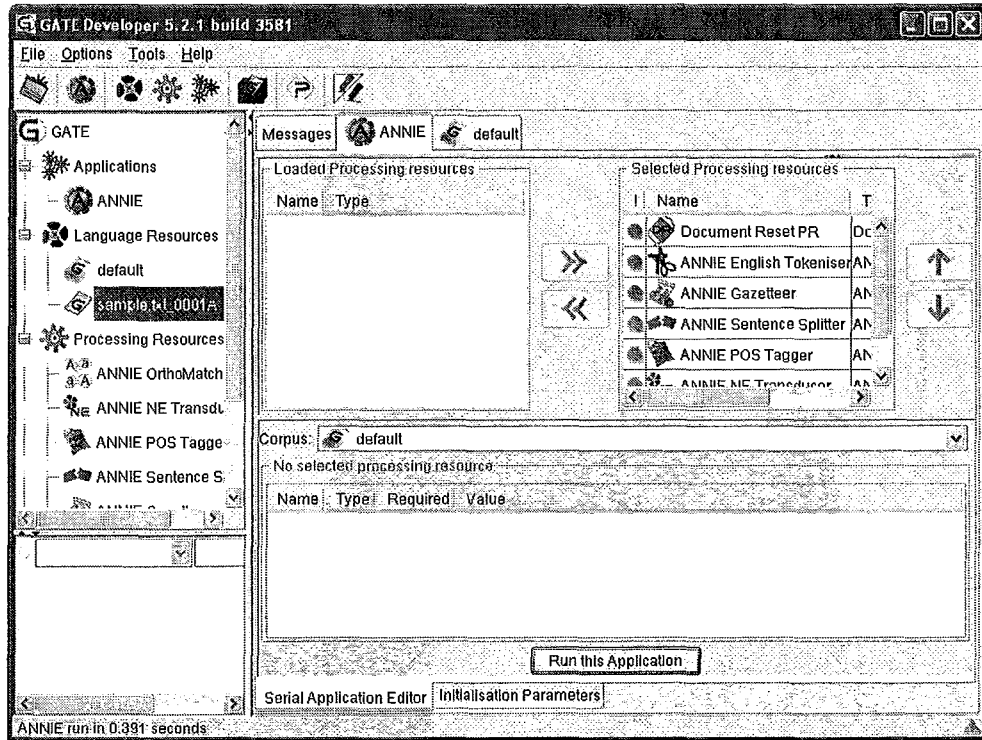


Figure 24 – Running ANNIE in GATE

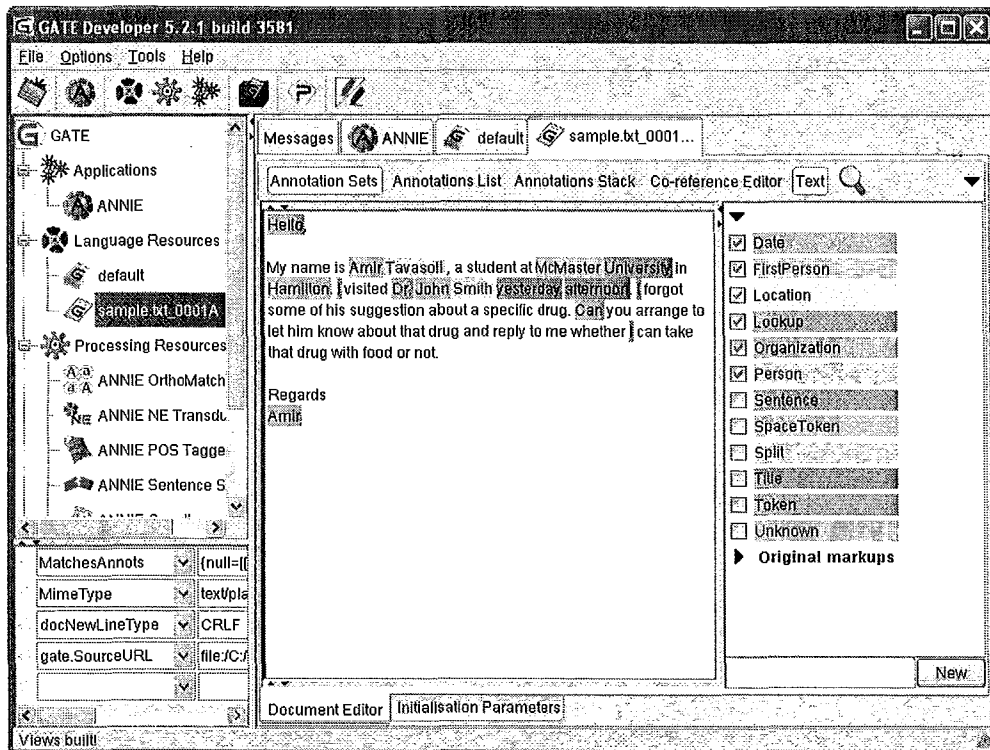


Figure 25 – GATE Finding Private Information

After finding this private information, the final step is to eliminate it from the entire corpus. A very simple solution was to run a simple Perl[85] script as follows to clear private information from the document:

```
#!/usr/bin/perl

@file_list = `ls inputxmls/`;

foreach $file(@file_list) {
#   print " => $file\n";
  open(XMLFILE, "inputxmls/$file") || die("Cannot open $file!");

  $output_filename = "cleanedxmls/$file";
  open(OUTFILE, ">> cleanedxmls/$file");

  #find "" patterns and replace it with ""
  while($_ = <XMLFILE>)
  {
    s/Amir Tavasoli/XXX/g;
    s/Amir/XXX/g;
    s/Tavasoli/XXX/g;
    s/McMaster/XXX/g;

    print OUTFILE ;
  }
}
```

```
}  
close(XMLFILE) || die("Cannot close $file!");  
close(OUTFILE) || die("Cannot close cleanedxmls/$file!");  
}
```

Appendix IV

Table of conflicts between message triage levels chosen by the two nurses. Note that the nn's that appear are actually carriage returns (keyboard "enters").

Subject	Message Content	Triage Level 1	Triage Level 2	Triage Level Final	Reason
Blood Pressure Monitor	Today I took my blood pressure and it seemed extremely high. I just wanted to put on record that this reading was done using my personal blood pressure monitor. At the intake meeting it was determined that my monitor needed the calibration checked. I am in the process of purchasing a new blood pressure monitor but did not want to delay starting the study so used my monitor and cuff.	1	2	1	Same as nurse 1 opinion, "Even if the BP reading was taken with a not well calibrated monitor under the circumstances of high BP reading is better to assume this an make sure patient is stable, immediate action should be taken."
Re: Cardiovascular risk factor survey	Hi n nMy replies to those questions are: 35 b 36 b 37 a. n nHope that finishes it. Thanks. N nxxx	2	3	3	It is just a survey and it does not need to be answered in 24 hours.
Re: BP readings on week 13	I was just wondering when the survey ends. Thanks xxx	2	3	3	It is just a survey and it does not need to be answered in 24 hours.

Re: BP readings on week 13	I was just wondering when the survey ends. Thanks xxx	2	4	4	Duplicate
Re: Cardiovascular risk factor survey	Hi n nMy replies to those questions are: 35 b 36 b 37 a. n nHope that finishes it. Thanks. N nxxx	2	4	4	Duplicate
Re: BP reading on week 19	Hi my blood pressure readings have bee pretty well the same. As usual I keep forgetting to mail them on to you. nI must apologise for missing the meeting last night. We had a bit of a family emergency spent the day in Burlington and I completely forgot about the meeting. nSorry about that. Nxxx	2	4	4	Duplicate
Holiday	I will be away from June 28-July 2 nd . Don't know if i can do the readings while away. Thanks xxx	2	3	3	Answering him/her is not very important in 24 hours, it just gives us some information about his trip
Holiday	I will be away from June 28-July 2 nd . Don't know if i can do the readings while away. Thanks xxx	2	4	4	Duplicate
Re: Cardiovascular risk factor survey	Why do I keep getting the messages to complete the surveys? I have done them more than once already. If you get this message please reply. Thank you. Nxxx	2	3	3	Agree with nurse comment "Even if is not BP related message, patient seems concern about this issue. Therefore I believe should be responded within 24 hours"

Re: Action plan Survey 3 reminder	Hi n I've seen the cardiovascular risk profile name under documents but when I click it nothing happens. I can not open it. I've tried that on different days but it hasn't opened so far. Sorry! N nxxx	2	4	4	Duplicate
Re: Allied Health Care Support	My only question is how come they haven't put in the medications that I use yet. Xxx	2	3	2	Seems better to answer this within 24 hours
help	Having problems with the mybp page again. Cannot get into the other system or use the calendar in it. So in this system the mybp into is missing. nCan you help?	2	3	2	Agree with nurse comment "There is technical problem relating to BP entry that I think should be fix within 24 hrs since patient is concern and willing to enter BP readings"
Re: BP reading on week 4	I'm a bit disappointed that my blood pressure is higher this morning. The downward trends have been vert encouraging. I'm thinking aobut what I ate yesterday which included: n- most of a 100 g package of Crispy Tortillaz chips so looks like about 900mg of sodium n- several more cups of coffee with caffeine than usual n- McDonalds chicken snack wrap for lunch n- less water than usual maybe only 2 cups n- steak tartar on a bun which probably was well-salted. nSo my food choices were not great	2	3	2	He/She seems disappointed and better to be encouraged fast.

	yesterday. And my "steps walked" were lower than usual the last 2 days especially yesterday. Today is a new day.				
Re: Exercise diary reminder	Sorry I made a schedule as you sent. I thought it was a sample. I will try to fill it in from what I recall. N nxxx	2	3	3	72 hours should be enough
my appointment on the 7 th	I would like to move the apt time from June 7 th to another date. If I can get the phone no of Project Support I would appreciate or have her call me n nThanking you nxxx.	3	2	2	Better to be dealt in 24 hours
my appointment on the 7 th	I would like to move the apt time from June 7 th to another date. If I can get the phone no of Project Support I would appreciate or have her call me n nThanking you . nxxx.	3	4	4	Duplicate
Re: BP reading on week 14	Yes I do remember that though I am not sure how to lower it just like that. Also yes my physician does know this and they are changing my medicine presently to see what can be done. Work continues on this! nUntil the next note. N nxxx	3	4	4	Duplicate

change of times for xxx	HI for XXX RE; appointment STONE CHURCH MED CNTR TUE JULY 7 for XXX 3.30pm family matters have come to unable to keep n aptm could we please book another one sorry for any 83nterference n XXX xxx-xxx-xxxx THANK YOU!!	3	2	2	Better to be dealt in 24 hours
Re: Action plan Survey first reminder	Hi n nI clicked my BP Action plan. Nothing happens. I apparently can't open my action plan. An inactive action plan. Ironi. nLet me know how I can activate that please. Thanks. N nI'll be gone most of the day but will be able to do that tonight after 10:00. N nxxx	3	4	4	Duplicate
bp	Is it always best to take my bp after sitting quietly or before doing any walking/housework etc.?	3	2	2	Quiet important to be answered in 24 hours
Re: Action plan Survey 3 reminder	Hi n nThanks. Now I found that. First thing it suggests is that I should make a print out of my blood pressure risk profile. Where will I find that? N nxxx	3	4	4	Duplicate
Re: Your Personal Action Plan and Exercise Diary	Hi n nThank you for the Exercise Diary. I'll try using that. I tried to open the Personal Action Plan and got a statement that it could not do certain parts; all it gave me was about ten lines of info. Was there more? N nxxx	3	4	4	Duplicate

Re: BP reading on week 7	Hi n nUpdate on my medicine: n n- Take off Atacand n nAdd: Apo Metoprolol tab 50 mg take 1/2 tabs PO BID for 30 days Qty: 30 Repeats: 0 Dr. XXX n nWe'll see what this will do. N nI can't figure out the bloodpressure business. They took it at the Health Centre. Result: 120/70 Is this the (new; your!) machine that gives me the high readings or is it something else? I am not accusing you just can't figure it. N nxxx	3	2	2	Better to be dealt in 24 hours
bp	Should I always be taking my bp on my right arm? N nThanks n nxxx	3	2	2	Better to be dealt in 24 hours
Re: BP reminder 1 on week 13	Hi n nI have filled in my Action Plan at least for April then I thought I had filed it by closing it. The next time I wanted to add to it there was nothing... How do I save this? Thanks. N nxxx	3	4	4	Duplicate
Re: Cardiovascular risk factor survey	Hi n nI get to answer three questions I suppose the three I apparently didn't answer the first time. It then tells me I can't finish the survey. Nxxx	3	4	4	Duplicate
Re: Action plan Survey first reminder	Hi n nI clicked my BP Action plan. Nothing happens. I apparently can't open my action plan. An inactive action plan. Ironic. nLet me know how I can activate that please. Thanks. N nI'll be gone most of the day but will be able to do that tonight after 10:00. N nxxx	4	3	3	Duplicate

<p>Re: BP reading on week 4</p>	<p>I've been pleased with my progress and choices over the past week. I've been able to log most or all of my food intake on most days. I have mostly been meeting my objectives for lots of fruits & vegetables low-fat foods and water. My weight is finally dropping a couple of pounds. I've been walking 6 500 – 10 000 steps a day over the past week. And my blood pressure is coming down a few points. Now I'm looking forward to seeing it in the target range. Thanks for this program which is giving me a way to measure my progress especially in such a visible tangible way as the BP chart.</p> <p>xxx n</p>	<p>4</p>	<p>3</p>	<p>3</p>	<p>Duplicate</p>
---------------------------------	---	----------	----------	----------	------------------

Table 22 - Resolving Differences Between Message Triage Levels Chosen by Two Nurses

The following table contains messages that were simulated to accommodate the shortage of level 0 and level 1 messages.

Number	Message Content	Triage Level
1	Hello, From a few hours ago I have a sever chest pain and I feel dizzy. There is something wrong with my blood pressure device so I don't know what is my situation right know, I just wanted to know what should I do know. Regards xxx	0
2	Hi, Last night when I wanted to go to sleep I had bad nausea and I woke up in the middle of night vomiting badly. Now I have just wake up from sleep and I can't see the things around clearly. I took my blood pressure and it was pretty high. I wanted to know if you can suggest anything to me. Thanks, XXX	0
3	Hi, I took my blood pressure right know and supperisingly I have a pretty elevated blood pressure. My blood pressure is know 160/100 . I also have a bad headache. I wanted to know if anything special I need to do. Thank You, XXX	0
4	Hello, I am a little bit nurvous. I feel very bad headache and chestpain. Yesterday my blood pressure was pretty high. It was around 160/130.I am alone in the house and I think I need to go to a hospital but I wanted to make sure if I really need to do so. Regards, XXX	0
5	Today is my birthday. Even thou I was told not to drink any alchohol. Because my friends insisted I had some drinks. Now I have a light headache with nausea. I feel like vomiting. I don't know whether it's a normal hang over or something sever. Please let me know. Thanks, XXX	0
6	I did some light exercise today, but after a few minutes I had shortness in breath.I have a some chestpain and headache also. I took my blood pressure and it was exteremly high. I tried to call my doctor but he was not available. Is there anything that I need to do? Thanks, XXX	0
7	My childeren went to a trip yesterday and I am alone in the house. I feel shaky and I have headache. My stomach is bothering me also. I cannot take my blood pressure my self and I cannot talk to my nurse to help me. Can you help me please with my situation. Thanks, XXX	0

8	I took my blood pressure in the morning and it was really high. Then I took my medicine as my doctor told me. After a few hours my blood pressure is still elevated. I have problems eating and I am alone in the house. Can you tell me what should I do? Thank You, XXX	0
9	Its midnight and I have problems sleeping. I have headache and when I wanted to walk to other room I fainted. My doctor and my nurse are not answering my call. I feel very bad and I wanted to know if there is anything I can do to feel better. Thanks, XXX	0
10	This morning when I woke up I couldn't find my blood pressure device. My blood pressure records was unsteady during the past few weeks. This morning I have some blurred vision and I have problems typing this message. Do you have any suggestions, Thanks, XXX	0
11	I stayed awake fully last night. I needed to work on my project. But this morning I have a very bad feeling. I feel some chestpain and I have problems walking fast. I'm in the office and I don't have any access to my blood pressure device. I took the medication but the feeling still remains. Can you help me? Yours, XXX	0
12	My blood pressure is usually high but since this morning it has been increasing rapidly. I cannot breathe very well now and I feel pain in my chest. What should I do?	0
13	My friend has checked my blood pressure and it was 150/92. I feel dizzy and cold. I never had this problem in my life so do you have any suggestions to prevent this disorder and if it happened again what should I do?	0
14	Hello nI check my blood pressure every day and it is usually 150/80 mmHG, but it's more than a week now that it stays around 175/80. Should I be concerned about and see a doctor?	0
15	I feel awful today. I had sleeping problems last night. I checked my blood pressure today and it was very high. I just wanted to let you know about my situation.	0
16	I am in a very bad situation today. My blood pressure is really high. I don't know what is the best thing to do. I wanted to ask if there is any suggestions from you.	0

17	I had a really bad day today. When I was walking to my job, I felt a little bit woozy. Then I started to have a headache in the afternoon. Now I have some chest pain and my blood pressure is elevated extremely. I was just wondering what is the best action to take. XXX	0
18	I don't know what's happening with my blood pressure device. From yesterday the device started to show very contradicting numbers. I feel dizzy and I have problem with my eyesight. But the device does not show a credible number. Can you help me with my blood pressure readings? Thanks, XXX	0
19	My blood pressure reading is really high this morning. It is about 170/130. I have been told to call my doctor in these situations but he is not available now. I took my medication but blood pressure is still elevated. I just wanted to report my blood pressure.	0
20	I have some stomach problems. I don't know whether they relate to hypertension or the food. I also feel dizzy and have some headache. I just wanted your idea about it. Thank You, XXX.	0
21	I have problems with my blood pressure device. It shows funny numbers. I feel almost ok despite a light headache. Can you help me with my device?	1
22	This morning I got my blood pressure and it was pretty elevated. My blood pressure is 140/130. I have some headache and stomach ache. I wanted to know if everything is ok. Thank You, XXX	1
23	Dear XXX, I checked my device and I mentioned that my blood pressure was higher than what I entered. It was about 150/130 instead 130/90. Is there anything I need to change in my action plan? Thanks, XXX	1
24	I have issues using myBP. Instead 140/120 I entered 120/90 in my records and I don't know how to change it. I don't know if it has any effects on my action plan. And note that I have a slight headache and my eyesight is a little bit blurry this morning.	1

25	Last night I needed to wake up 3 times and take some cold water. I feel a little bit dizzy this morning. I called doctor and he said the everything is ok and your blood pressure will be controlled in a few days. My blood pressure is 140/110 and I cannot enter it on my records. Can you help me please.	1
26	As of this morning I felt a little bit dizzy. I never felt this way before and this is the first time during the study that I have this feeling. I have a slight headache and I wanted to know if I am doing anything wrong.	1
27	Just add my recent blood pressure to my record. It is 150/130 and it is highr than yesterday which was 130/110. Thank You, XXX	1
28	I couldn't keep up with my action plan and I had pretty heavy pizza yesterday. I have a slight nausea and feel some headache. My blood pressure is higher than yesterday. Is everything is ok?	1
29	I had some honey wings yesterday even thou I was banned to have this kind of heavy food. I feel a little bit dizzy and feel like vomiting. Is there anything I need to do? Thanks, XXX	1
30	Last week I was going with my action plan and everything was ok. However, from this morning I feel a slight headache and my blood pressure has been elevated. I don't know what I did wrong. Can you help me.	1
31	I have a little bit dizzy feeling through last two or three days. It does not bother me that much but I wanted to make sure that this headache is normal, thank you, XXX	1
32	I cannot stop the headache started 2 days ago. I can take my normal actions but the headache won't go away. I haven't tested my blood pressure in these two days. Can you tell me what is the best thing to do?	1
33	I have heard a very bad news this morning. I am short on breath. As the times goes I feel better and better but I wanted to make sure that everything is ok. Thanks, XXX	1
34	I think I need to talk to a doctor about my problem. I try to follow my action plan but I still have high blood pressure and it does not go down. I have a slight headache as of this morning. Thanks, XXX	1

35	Can you help me with my action plan. I follow it fairly but I seem to have a little bit of headache and feel weak-kneed also. Thanks, XXX	1
36	Everything was ok last week but from this week my blood pressure device is showing elevated numbers. My condition is almost ok despite some unsteady headaches. Can you help me with my situation.	1
37	I was reporting my blood pressure using my device for about month but yesterday my daughter came to meet me and she figured out that I'm doing it in wrong way. After that I realised that my blood pressure is higher than I expected. Is there anything I need to change. Thank you, XXX	1
38	When I woke up yesterday, I had a slight headache. I ignored it for the day and by the afternoon it seem to be better but again today's morning the headache came back. My blood pressure seem to be elevated also. What I need to do?	1
39	I can't take the headache that started a few hours ago. I took my blood pressure it was quiet normal. Can you help me with my headache.	1
40	I woke up last night during the night I felt a slight headache, I took my blood pressure and it was pretty high. Then took my medication and I was able to sleep. In the morning when I took my blood pressure it seems to be quiet ok. Now tonight I again feel the headache. Can you help me with my situation. Regards, XXX	1

Table 23 – Messages Simulated to Overcome Lack of Level 0 and Level 1 Messages

Appendix V

This is the implementation of the classifiers mentioned in this work. This code was implemented using Java and Eclipse IDE (www.eclipse.org).

The first part of the code includes the added function *KnnClassifier Class*, used to determine the neighborhood of a given document.

```

public ArrayList<ScoredObject<Pair<String, Integer>>>
returnNeighbours(E in,
    String Docname, Integer numNeighbours, int type) {
    Map<String,? extends Number> featureMap
        = mFeatureExtractor.features(in);
    SparseFloatVectorWithDocname inputVector
        = Features.toVectorWithDocnameAddSymbols(featureMap,
            mFeatureSymbolTable,
            Integer.MAX_VALUE-1,
            false,
            Docname);

    EuclideanDistance ed = null;
    if(type == 0) {
        ed = new EuclideanDistance();
    }

    CosineDistance cd = null;
    if(type == 1 || type == 2) {
        cd = new CosineDistance();
    }

    TaxicabDistance td = new TaxicabDistance();
    MinkowskiDistance md = new MinkowskiDistance(3);

    BoundedPriorityQueue<ScoredObject<Pair<String, Integer>>> queue
        = new
BoundedPriorityQueue<ScoredObject<Pair<String, Integer>>>(
    ScoredObject.comparator(), numNeighbours /*mK*/);
    for (int i = 0; i < mTrainingCategories.size(); ++i) {
        Integer catId = mTrainingCategories.get(i);
        SparseFloatVectorWithDocname trainingVector =
            mTrainingVectors.get(i);

        ////////// Different ways to build neighborhood
        double score = 0.0;
        //Euclidean Distance
        if(type == 0) {
            score = (1.0/(1.0
                +ed.distance(inputVector, trainingVector)));
        }
    }
}

```

```

//Cosine Distance
if(type == 1) {
    score = (1.0/(1.0
        +cd.distance(inputVector, trainingVector)));
}

if(type == 2) {
    score = cd.proximity(inputVector, trainingVector);
}

//Taxicab Distance
if(type==3) {
    score = (1.0/(1.0+
        td.distance(inputVector, trainingVector)));
}
//Minkowski Distance
if(type==4) {
    score = (1.0/(1.0+
        md.distance(inputVector, trainingVector)));
}
//build a vector of biggest proximity
queue.offer( new ScoredObject<Pair<String,Integer>>(
new Pair ( trainingVector.DocumentName , catId ),score));
}

ArrayList<ScoredObject<Pair<String,Integer>>> output
= new ArrayList<ScoredObject<Pair<String,Integer>>>();

while(!queue.isEmpty()) {
output.add(queue.remove());
}

return output;
}

public Vector returnVector(E in, String Docname) {
    Map<String,? extends Number> featureMap
        = mFeatureExtractor.features(in);
return (Vector)Features.toVectorWithDocnameAddSymbols(featureMap,
        mFeatureSymbolTable,
        Integer.MAX_VALUE-1,
        false,
        Docname);
}

```

There are some minor edits in other parts of the class to save the document name (not mentioned here).

Some minor edits were used on the Features class to allow this class to retrieve the names of documents. Document names were used to calculate the neighborhood of a given document.

```

public static SparseFloatVectorWithDocname
    toVectorWithDocnameAddSymbols (Map<String, ? extends Number>
featureVector,
                                SymbolTable table,
                                int numDimensions,
                                boolean addIntercept,
                                String Docname) {
    int size = (featureVector.size() * 3) / 2;
    Map<Integer, Number> vectorMap
        = new HashMap<Integer, Number>(size);
    for (Map.Entry<String, ? extends Number> entry
        : featureVector.entrySet()) {
        String feature = entry.getKey();
        Number val = entry.getValue();
        int id = table.getOrAddSymbol(feature);
        vectorMap.put(Integer.valueOf(id), val);
    }
    if (addIntercept)
        vectorMap.put(Integer.valueOf(0), 1.0);
    return new SparseFloatVectorWithDocname (vectorMap,
numDimensions, Docname);
}

```

A new class was added to test the documents and group them into test or train groups

```

package classifiers;

import java.io.File;

enum FileState {Test, Train};

public class FileBinayType {
    public File file;
    public FileState state;

    FileBinayType(File f, FileState s)
    {
        file = f;
        state = s;
    }
}

```

The main part of the code, which includes all the work with classifiers and LingPipe Library, is followed. It is worth mentioning that the codes for individual classifiers have been adapted from [1].

```

package classifiers;

import java.io.*;
import java.util.*;

```

```

import com.aliasi.classify.*;
import com.aliasi.matrix.CosineDistance;
import com.aliasi.matrix.EuclideanDistance;
import com.aliasi.matrix.MinkowskiDistance;
import com.aliasi.matrix.TaxicabDistance;
import com.aliasi.spell.TfIdfDistance;
import com.aliasi.tokenizer.*;
import com.aliasi.util.Files;
import com.aliasi.util.BoundedPriorityQueue;
import com.aliasi.util.Pair;
import com.aliasi.util.ScoredObject;

public class compare {
    private static String FS =
System.getProperty("file.separator");
    private static File ORIGINAL_DIR = new File("./original");
    private static float PRCNT_TEST = (float) 0.2;
    private static String modelFileDir =
        ORIGINAL_DIR + FS + "training_files" + FS;
    private static String[] CATEGORIES =
{"level0", "level1", "level2", "level3"};
    private static int NUM_RUNS = 100;
    ///////////////////////////////////////////////////////////////////
    ///////////////////////////////////////////////////////////////////
    //////////////// New Classifier
    ///////////////////////////////////////////////////////////////////
    ///////////////////////////////////////////////////////////////////
public static void trainNewClassifier(Vector<Vector> mainVector,
    String lmModelFile, String knnModelFile,
    String navieBayesModelFile, String tfidfModelFile,
    int K_SIZE, int NGRAM_SIZE) {

    //Language Model Classifier
    DynamicLMClassifier lmclassifier =
    DynamicLMClassifier.createNGramBoundary(CATEGORIES,
    NGRAM_SIZE);

    //Knn Classifier
    KnnClassifier<CharSequence> knnclassifier=
    new KnnClassifier<CharSequence> (
    new TokenFeatureExtractor(
    new IndoEuropeanTokenizerFactory()), K_SIZE,
    new EuclideanDistance() );

    //Navie Bayes Classifier
    NaiveBayesClassifier naivebayesclassifier =
    new NaiveBayesClassifier(CATEGORIES,
    new IndoEuropeanTokenizerFactory());

    //TF-IDF Classifier
    TfIdfClassifierTrainer<CharSequence> tfidfclassifier = new
    TfIdfClassifierTrainer<CharSequence> (

```

```

    new TokenFeatureExtractor(
    new IndoEuropeanTokenizerFactory()) );

try {
for(int i=0;i<CATEGORIES.length; i++ ) {
    Vector<FileBinayType> catVect = mainVector.get(i);

    for(FileBinayType tf : catVect) {
        if(tf.state != null && tf.state == FileState.Train) {
            String text = Files.readFromFile(tf.file,"UTF-8");

            lmclassifier.train(CATEGORIES[i], text, NGRAM_SIZE);
            knnclassifier.handle(new Classified(text,
                new Classification(CATEGORIES[i])),
                tf.file.getName());

            naivebayesclassifier.train(CATEGORIES[i], text,
                refinedText.length());

            tfidfclassifier.handle(new Classified(text,
                new Classification(CATEGORIES[i])));
        }
    }
ObjectOutputStream os = new ObjectOutputStream(
    new FileOutputStream(lmModelFile));
lmclassifier.compileTo(os);
os.close();

os = new ObjectOutputStream(
    new FileOutputStream(knnModelFile));
knnclassifier.compileTo(os);
os.close();

os = new ObjectOutputStream(
    new FileOutputStream(naiveBayesModelFile));
naivebayesclassifier.compileTo(os);
os.close();

os = new ObjectOutputStream(
    new FileOutputStream(tfidfModelFile));
tfidfclassifier.compileTo(os);
os.close();
} catch(IOException e) {
    e.printStackTrace();
}
}

//////////
public static ConfusionMatrix testDynamicClassifierSelection(
Vector<Vector> mainVector, String lmModelFile, String
knnModelFile,
String naiveBayesModelFile, String tfidfModelFile) {

```

```

ConfusionMatrix confMatrix =
    new ConfusionMatrix(CATEGORIES);
try {

//LM Classifier
ObjectInputStream oi = new ObjectInputStream(
    new FileInputStream(lmModelFile) );

LMClassifier lmCompiledClassifier =
    (LMClassifier) oi.readObject();
oi.close();

//Knn Classifier
oi = new ObjectInputStream(
    new FileInputStream(knnModelFile) );

KnnClassifier<CharSequence> knnCompiledClassifier =
    (KnnClassifier<CharSequence>) oi.readObject();
oi.close();

//Navie Bayes Classifier
oi = new ObjectInputStream(
    new FileInputStream(navieBayesModelFile) );

LMClassifier naiveBayesCompiledClassifier =
    (LMClassifier) oi.readObject();
oi.close();

//TF-IDF Classifier
oi = new ObjectInputStream(
    new FileInputStream(tfidfModelFile) );

BaseClassifier tfidfCompiledClassifier =
    (BaseClassifier) oi.readObject();
oi.close();

for(int i=0;i<CATEGORIES.length; i++ ) {
    Vector<FileBinayType> catVect = mainVector.get(i);
    String text;
    for(FileBinayType tf : catVect) {
        if(tf.state != null && tf.state == FileState.Test) {
            text = Files.readFromFile(tf.file,"UTF-8");

            //Find K nearest neighbor of the sample that we want
            to classify
            ArrayList<ScoredObject<Pair<String,Integer>>>
                ScoredNeighbors = knnCompiledClassifier.
returnNeighbours( (CharSequence) text, tf.file.getName(), 4, 0);

ConfusionMatrix localLMConfMatrix =
    new ConfusionMatrix(CATEGORIES);

```



```

ConfusionMatrix localKNNConfMatrix =
    new ConfusionMatrix(CATEGORIES);

ConfusionMatrix localNBConfMatrix =
    new ConfusionMatrix(CATEGORIES);

ConfusionMatrix localTFIDFConfMatrix =
    new ConfusionMatrix(CATEGORIES);

for(int p = 0;p < ScoredNeighbors.size();p++) {
    ScoredObject<Pair<String,Integer>> so =
        ScoredNeighbors.get(p);

    int localCatID = so.getObject().b();
    String docname = so.getObject().a();
    //Find how our classifiers classify that sample
    File subFile = new File(ORIGINAL_DIR + FS +
        CATEGORIES[localCatID] + FS + so.getObject().a());

    String subText = Files.readFromFile(
        subFile,"UTF-8");

    JointClassification lmjc =
        lmCompiledClassifier.classifyJoint(
            subText.toCharArray(), 0, subText.length());

    localLMConfMatrix.increment(
        CATEGORIES[localCatID],
        lmjc.bestCategory());

    ScoredClassification knnsc =(ScoredClassification)
        KnnCompiledClassifier.classify(subText);

    localKNNConfMatrix.increment(
        CATEGORIES[localCatID],
        knnsc.bestCategory());

    JointClassification naivebayesjc =
        naiveBayesCompiledClassifier.classifyJoint(
            subText.toCharArray(), 0, subText.length());

    localNBConfMatrix.increment(
        CATEGORIES[localCatID],
        naivebayesjc.bestCategory());

    ScoredClassification tfidfsc =
        (ScoredClassification)
        tfidfCompiledClassifier.classify(subText);

    localTFIDFConfMatrix.increment(

```

```

        CATEGORIES[localCatID],
        tfidfsc.bestCategory());
    }
    //Choose the classifier that does the best and put at
as output
    int bestLocalClassifier = findMax(
        localLMConfMatrix.totalAccuracy(),
        localKNNConfMatrix.totalAccuracy(),
        localNBConfMatrix.totalAccuracy(),
        localTFIDFConfMatrix.totalAccuracy());

    //find the classifier with minimum error

    //if LM is chosen
    if(bestLocalClassifier == 0) {
        JointClassification jc =
            lmCompiledClassifier.classifyJoint(
                text.toCharArray(), 0, text.length());

        confMatrix.increment(
            CATEGORIES[i], jc.bestCategory());
    //if Knn is chosen
    } else if(bestLocalClassifier == 1) {
        ScoredClassification classification =
            (ScoredClassification)
            knnCompiledClassifier.classify(text);
        confMatrix.increment(CATEGORIES[i],
            classification.bestCategory());
    //if Naive Bayes is chosen
    } else if(bestLocalClassifier == 2) {
        JointClassification jc =
            naiveBayesCompiledClassifier.classifyJoint(
                text.toCharArray(), 0,
                text.length());
        confMatrix.increment(
            CATEGORIES[i], jc.bestCategory());
    //if TF/IDF is chosen
    } else if(bestLocalClassifier == 3) {
        ScoredClassification classification =
            (ScoredClassification)
            tfidfCompiledClassifier.classify(text);
        confMatrix.increment(CATEGORIES[i],
            classification.bestCategory());
    }
    }
}
} catch(Exception e) {
    e.printStackTrace();
}

```

```

    return confMatrix;
}

////////////////////////////////////
// Adaptive classifier combination
////////////////////////////////////
public static ConfusionMatrix testAdaptiveClassifierComb(
Vector<Vector> mainVector, String lmModelFile,
String knnModelFile, String navieBayesModelFile, String
tfidfModelFile) {
    ConfusionMatrix confMatrix =
        new ConfusionMatrix(CATEGORIES);
    try {
        //LM Classifier
        ObjectInputStream oi = new ObjectInputStream(
            new FileInputStream(lmModelFile) );

        LMClassifier lmCompiledClassifier =
            (LMClassifier) oi.readObject();
        oi.close();

        //Knn Classifier
        oi = new ObjectInputStream(
            new FileInputStream(knnModelFile) );

        KnnClassifier<CharSequence> knnCompiledClassifier =
            (KnnClassifier<CharSequence>) oi.readObject();
        oi.close();

        //Navie Bayes Classifier
        oi = new ObjectInputStream(
            new FileInputStream(navieBayesModelFile) );

        LMClassifier naiveBayesCompiledClassifier =
            (LMClassifier) oi.readObject();
        oi.close();

        //TF-IDF Classifier
        oi = new ObjectInputStream(
            new FileInputStream(tfidfModelFile) );

        BaseClassifier tfidfCompiledClassifier =
            (BaseClassifier) oi.readObject();
        oi.close();

        for(int i=0;i<CATEGORIES.length; i++ ) {
            Vector<FileBinayType> catVect = mainVector.get(i);
            String text;
            for(FileBinayType tf : catVect) {
                if(tf.state != null && tf.state == FileState.Test) {
                    text = Files.readFromFile(tf.file,"UTF-8");
                }
            }
        }
    }
}

```

```

classifiers //an array that keeps results of different
String[] classesSelected = new String[4];

//classify new document to see which categories it
belongs to
JointClassification lmjc =
    lmCompiledClassifier.classifyJoint(
        text.toCharArray(), 0, text.length());

classesSelected[0] = lmjc.bestCategory();

ScoredClassification knnsc =(ScoredClassification)
    knnCompiledClassifier.classify(text);

classesSelected[1] = knnsc.bestCategory();

JointClassification naivebayesjc =
    naiveBayesCompiledClassifier.classifyJoint(
        text.toCharArray(), 0, text.length());

classesSelected[2] = naivebayesjc.bestCategory();

ScoredClassification tfidfsc = (ScoredClassification)
    tfidfCompiledClassifier.classify(text);

classesSelected[3] = tfidfsc.bestCategory();

//Find K nearest neighbor of the sample that we want
to classify
ArrayList<ScoredObject<Pair<String,Integer>>>
    ScoredNeighbors =
    knnCompiledClassifier.returnNeighbours(
        (CharSequence) text, tf.file.getName(), 4, 1);

double[] Wi = new double[ScoredNeighbors.size()];
double[] Acc = {0.0, 0.0, 0.0, 0.0};

CosineDistance cd = new CosineDistance();

com.aliasi.matrix.Vector inputVect =
    knnCompiledClassifier.returnVector(
        (CharSequence) text, tf.file.getName());

for(int s = 0;s < classesSelected.length;s++)
    for(int p = 0;p < ScoredNeighbors.size();p++) {
        ScoredObject<Pair<String,Integer>> so =
            ScoredNeighbors.get(p);

        int catID = so.getObject().b();
        String docname = so.getObject().a();
        //Open and read the neighbor file

```

```

File subFile = new File( ORIGINAL_DIR + FS +
    CATEGORIES[catID] + FS + docname);

String subText =
Files.readFromFile(subFile, "UTF-8");

com.aliasi.matrix.Vector subVect =
    knnCompiledClassifier.returnVector(
        (CharSequence) subText, docname);

Wi[p] = cd.distance(inputVect, subVect);

JointClassification locallmjc =
lmCompiledClassifier.classifyJoint(
    subText.toCharArray(), 0,
    subText.length());

Acc[s] +=
    Wi[p]*locallmjc.conditionalProbability(
        classesSelected[s]);

ScoredClassification localknsc =
    (ScoredClassification)
    knnCompiledClassifier.classify(subText);

Acc[s] += Wi[p]*ScoreToProbability(localknsc,
    catStr2Int(classesSelected[s]));

JointClassification localnaivebayesjc =
naiveBayesCompiledClassifier.classifyJoint(
    subText.toCharArray(), 0, subText.length());

Acc[s] +=
    Wi[p]*localnaivebayesjc.conditionalProbability(
        classesSelected[s]);

ScoredClassification localtfidfsc =
    (ScoredClassification)
    tfidfCompiledClassifier.classify(subText);

Acc[s] += Wi[p]*ScoreToProbability(localtfidfsc,
    catStr2Int(classesSelected[s]));
}

int bestLocalClassifier = findMax(Acc);

confMatrix.increment(CATEGORIES[i],
    classesSelected[bestLocalClassifier]);
}
}
} catch (Exception e) {

```

```

        e.printStackTrace();
    }
    return confMatrix;
}

////////////////////////////////////
// Simple Voting Combination
////////////////////////////////////
public static ConfusionMatrix testSimpleVotingCombClassifier(
    Vector<Vector> mainVector, String lmModelFile,
    String knnModelFile, String navieBayesModelFile, String
    tfidfModelFile) {
    ConfusionMatrix confMatrix =
        new ConfusionMatrix(CATEGORIES);
    try {
        //LM Classifier
        ObjectInputStream oi = new ObjectInputStream(
            new FileInputStream(lmModelFile) );

        LMClassifier lmCompiledClassifier =
            (LMClassifier) oi.readObject();
        oi.close();

        //Knn Classifier
        oi = new ObjectInputStream(
            new FileInputStream(knnModelFile) );

        KnnClassifier<CharSequence> knnCompiledClassifier =
            (KnnClassifier<CharSequence>) oi.readObject();
        oi.close();

        //Navie Bayes Classifier
        oi = new ObjectInputStream(
            new FileInputStream(navieBayesModelFile) );

        LMClassifier naiveBayesCompiledClassifier =
            (LMClassifier) oi.readObject();
        oi.close();

        //TF-IDF Classifier
        oi = new ObjectInputStream(
            new FileInputStream(tfidfModelFile) );

        BaseClassifier tfidfCompiledClassifier =
            (BaseClassifier) oi.readObject();
        oi.close();

        for(int i=0;i<CATEGORIES.length; i++ ) {
            Vector<FileBinayType> catVect = mainVector.get(i);
            String text;
            for(FileBinayType tf : catVect) {
                if(tf.state != null && tf.state == FileState.Test) {
                    text = Files.readFromFile(tf.file,"UTF-8");
                }
            }
        }
    }
}

```

```

//an array that keeps results of different classifiers
String[] classesSelected = new String[4];
double[] vote = new double[CATEGORIES.length];
for(int k=0;k<CATEGORIES.length;k++) {
    vote[k] = 0.0;
}
//classify new document to see which categories it
//belongs to
JointClassification lmjc =
    lmCompiledClassifier.classifyJoint(
        text.toCharArray(), 0, text.length());
classesSelected[0] = lmjc.bestCategory();
vote[ catStr2Int(classesSelected[0]) ] += 1.0 *
    lmjc.conditionalProbability(classesSelected[0]);

ScoredClassification knnsc = (ScoredClassification)
    knnCompiledClassifier.classify(text);
classesSelected[1] = knnsc.bestCategory();

vote[ catStr2Int(classesSelected[1]) ] += 1.0 *
    ScoreToProbability(knnsc,
        catStr2Int(classesSelected[1]));

JointClassification naivebayesjc =
    naiveBayesCompiledClassifier.classifyJoint
        text.toCharArray(), 0, text.length());
classesSelected[2] = naivebayesjc.bestCategory();

vote[ catStr2Int(classesSelected[2]) ] += 1.0 *
    naivebayesjc.conditionalProbability
        (classesSelected[2]);

ScoredClassification tfidfsc =(ScoredClassification)
    tfidfCompiledClassifier.classify(text);
classesSelected[3] = tfidfsc.bestCategory();

vote[ catStr2Int(classesSelected[3]) ] += 1.0 *
    ScoreToProbability(knnsc,
        catStr2Int(classesSelected[3]));

//See which class been chosen the most

int bestLocalClassifier = findMax(vote);

confMatrix.increment(CATEGORIES[i],
    CATEGORIES[bestLocalClassifier]);
}
}
} catch(Exception e) {
    e.printStackTrace();
}

```

```

    return confMatrix;
}

////////////////////////////////////
////////////////////////////////////
////// New Combination Algorithm
////////////////////////////////////
public static ConfusionMatrix testNewCombClassifier(
Vector<Vector> mainVector,String lmModelFile, String knnModelFile,
String navieBayesModelFile, String tfidfModelFile) {

    ConfusionMatrix confMatrix =
        new ConfusionMatrix(CATEGORIES);

    try {

        //LM Classifier
        ObjectInputStream oi = new ObjectInputStream(
            new FileInputStream(lmModelFile) );

        LMClassifier lmCompiledClassifier =
            (LMClassifier) oi.readObject();
        oi.close();

        //Knn Classifier
        oi = new ObjectInputStream(
            new FileInputStream(knnModelFile) );

        KnnClassifier<CharSequence> knnCompiledClassifier =
            (KnnClassifier<CharSequence>) oi.readObject();
        oi.close();

        //Navie Bayes Classifier
        oi = new ObjectInputStream(
            new FileInputStream(navieBayesModelFile) );

        LMClassifier naiveBayesCompiledClassifier =
            (LMClassifier) oi.readObject();
        oi.close();

        //TF-IDF Classifier
        oi = new ObjectInputStream(
            new FileInputStream(tfidfModelFile) );

        BaseClassifier tfidfCompiledClassifier =
            (BaseClassifier) oi.readObject();
        oi.close();

        for(int i=0;i<CATEGORIES.length; i++ ) {
            Vector<FileBinayType> catVect = mainVector.get(i);
            String text;
            for(FileBinayType tf : catVect) {
                if(tf.state != null && tf.state == FileState.Test) {

```



```

text = Files.readFromFile(tf.file, "UTF-8");

double[] Score = new double[CATEGORIES.length];
//an array that keeps results of different
//classifiers
String[] classesSelected = new String[4];
double[] vote = new double[CATEGORIES.length];
double total = 0.0;
for(int k=0;k<CATEGORIES.length;k++) {
    vote[k] = 0.0;
    Score[k] = 0.0;
}

//classify new document to see which categories it
//belongs to
JointClassification lmjc =
    lmCompiledClassifier.classifyJoint(
        text.toCharArray(), 0, text.length());
classesSelected[0] = lmjc.bestCategory();
vote[ catStr2Int(classesSelected[0]) ] += 1.0 *
    lmjc.conditionalProbability(classesSelected[0]);

ScoredClassification knnsc =
    (ScoredClassification)
        knnCompiledClassifier.classify(text);
classesSelected[1] = knnsc.bestCategory();
vote[ catStr2Int(classesSelected[1]) ] += 1.0 *
    ScoreToProbability(knnsc,
        catStr2Int(classesSelected[1]));

JointClassification naivebayesjc =
    naiveBayesCompiledClassifier.classifyJoint(
        text.toCharArray(), 0, text.length());
classesSelected[2] = naivebayesjc.bestCategory();
vote[ catStr2Int(classesSelected[2]) ] += 1.0 *
    naivebayesjc
        .conditionalProbability(classesSelected[2]);

ScoredClassification tfidfsc = (ScoredClassification)
    tfidfCompiledClassifier.classify(text);
classesSelected[3] = tfidfsc.bestCategory();
vote[ catStr2Int(classesSelected[3]) ] += 1.0 *
    ScoreToProbability(knnsc,
        catStr2Int(classesSelected[3]));

////////////////////////////////////////
//Find K nearest neighbor of the sample that we want
to classify
ArrayList<ScoredObject<Pair
    <String, Integer>>>
    ScoredNeighbors = null;
ScoredNeighbors =

```

```

        knnCompiledClassifier.returnNeighbours
            ((CharSequence) text, tf.file.getName(), 9, 1);

    double[] Wi = new double[ScoredNeighbors.size()];
    double[] Acc = {0.0, 0.0, 0.0, 0.0};

    CosineDistance cd = new CosineDistance();

    com.aliasi.matrix.Vector inputVect =
        knnCompiledClassifier.returnVector(
            (CharSequence) text, tf.file.getName());

    ArrayList<ScoredObject<Pair
        <String,Integer>>> toRemove
        = new ArrayList<ScoredObject<Pair
            <String,Integer>>>();

    for(int p = 0; p < ScoredNeighbors.size(); p++) {
        ScoredObject<Pair<String,Integer>> so =
            ScoredNeighbors.get(p);
        int catID = so.getObject().b();
        String docname = so.getObject().a();
        //Open and read the neighbor file
        File subFile = new File( ORIGINAL_DIR + FS +
            CATEGORIES[catID] + FS + docname);

        String subText =
            Files.readFromFile(subFile,"UTF-8");

        com.aliasi.matrix.Vector subVect =
            knnCompiledClassifier.returnVector(
                (CharSequence) subText, docname);

        if(cd.proximity(inputVect,subVect) < .70) {
            toRemove.add(so);
        }
    }

    if(ScoredNeighbors.size() == toRemove.size()) {
        toRemove.remove(0);
        toRemove.remove(0);
    } else if(ScoredNeighbors.size() ==
        (toRemove.size()+1)) {
        toRemove.remove(0);
    }

    ScoredNeighbors.removeAll(toRemove);

    for(int s = 0; s < classesSelected.length; s++)
        for(int p = 0; p < ScoredNeighbors.size(); p++) {
            ScoredObject<Pair<String,Integer>> so =
                ScoredNeighbors.get(p);

```

```

int catID = so.getObject().b();
String docname = so.getObject().a();
//Open and read the neighbor file
File subFile = new File( ORIGINAL_DIR + FS +
    CATEGORIES[catID] + FS + docname);

String subText =
    Files.readFromFile(subFile,"UTF-8");

com.aliasi.matrix.Vector subVect =
    knnCompiledClassifier.returnVector(
        (CharSequence) subText, docname);

Wi[p] = cd.proximity(inputVect,subVect);

JointClassification locallmjc =
    lmCompiledClassifier.classifyJoint(
        subText.toCharArray(), 0, subText.length());

Acc[s] +=
    Wi[p]*locallmjc.conditionalProbability(
        classesSelected[s]);

ScoredClassification localknsc =
    (ScoredClassification)
    knnCompiledClassifier.classify(subText);

Acc[s] += Wi[p]*ScoreToProbability(localknsc,
    catStr2Int(classesSelected[s]));

JointClassification localnaivebayesjc =
    naiveBayesCompiledClassifier.classifyJoint(
        subText.toCharArray(), 0, subText.length());

Acc[s] += Wi[p]*localnaivebayesjc.
    conditionalProbability(
        classesSelected[s]);

ScoredClassification localtfidfsc =
    (ScoredClassification)
    tfidfCompiledClassifier.classify(subText);

Acc[s] +=
    Wi[p]*ScoreToProbability(localtfidfsc,
        catStr2Int(classesSelected[s]));

}
//Select the classifier with highest Score
int bestClassifier = findMax(Acc);

confMatrix.increment(CATEGORIES[i],
    classesSelected[bestClassifier]);
}

```

```

    }
}
} catch(Exception e) {
    e.printStackTrace();
}
return confMatrix;
}

//////////////////////////////////////
//////////////////////////////////////
////////////////////////////////////// Language Model Classifier
//////////////////////////////////////
//////////////////////////////////////
public static void trainLM(Vector<Vector> mainVector,
String modelFile, int NGRAM_SIZE) {
    DynamicLMClassifier classifier =
        DynamicLMClassifier.createNGramBoundary(
            CATEGORIES, NGRAM_SIZE);
    try {
        for(int i=0;i<CATEGORIES.length; i++ ) {
            Vector<FileBinayType> catVect = mainVector.get(i);
            for(FileBinayType tf : catVect) {
                if(tf.state != null && tf.state == FileState.Train) {
                    String text = Files.readFromFile(tf.file,"UTF-8");
                    classifier.train(CATEGORIES[i], text, NGRAM_SIZE);
                }
            }
            ObjectOutputStream os = new ObjectOutputStream(
                new FileOutputStream(modelFile));
            classifier.compileTo(os);
            os.close();
        }
    } catch(IOException e) {
        e.printStackTrace();
    }
}

//////////////////////////////////////
public static ConfusionMatrix testLM(Vector<Vector> mainVector,
String modelFile) {
    ConfusionMatrix confMatrix =
        new ConfusionMatrix(CATEGORIES);
    try {
        ObjectInputStream oi = new ObjectInputStream(
            new FileInputStream(modelFile) );
        LMClassifier compiledClassifier =
            (LMClassifier) oi.readObject();
        oi.close();

        for(int i=0;i<CATEGORIES.length; i++ ) {
            Vector<FileBinayType> catVect = mainVector.get(i);
            String text;

```

```

    for(FileBinayType tf : catVect) {
        if(tf.state != null && tf.state == FileState.Test) {
            text = Files.readFromFile(tf.file,"UTF-8");
            JointClassification jc =
                compiledClassifier.classifyJoint(
                    text.toCharArray(), 0, text.length());
            confMatrix.increment(
                CATEGORIES[i], jc.bestCategory());
        }
    }
} catch(Exception e) {
    e.printStackTrace();
}
return confMatrix;
}

////////////////////////////////////
////////////////////////////////////
//////////////////////////////////// Knn Classifier
////////////////////////////////////
////////////////////////////////////
public static void trainKnn(Vector<Vector> mainVector, String
modelFile, int K_SIZE) {
    KnnClassifier<CharSequence> classifier=
        new KnnClassifier<CharSequence> (
            new TokenFeatureExtractor(
                new IndoEuropeanTokenizerFactory()), K_SIZE,
            new EuclideanDistance());
    try {
        for(int i=0;i<CATEGORIES.length; i++ ) {
            Vector<FileBinayType> catVect = mainVector.get(i);
            for(FileBinayType tf : catVect) {
                if(tf.state != null && tf.state == FileState.Train) {
                    String text = Files.readFromFile(tf.file,"UTF-8");
                    classifier.handle(new Classified(text,
                        new Classification(CATEGORIES[i])));
                }
            }

            ObjectOutputStream os = new ObjectOutputStream(
                new FileOutputStream(modelFile));
            classifier.compileTo(os);
            os.close();
        }
    } catch(IOException e) {
        e.printStackTrace();
    }
}

////////////////////////////////////
public static ConfusionMatrix testKnn(Vector<Vector> mainVector,
String modelFile) {

```

```

ConfusionMatrix confMatrix =
    new ConfusionMatrix(CATEGORIES);

try {
ObjectInputStream oi = new ObjectInputStream(
    new FileInputStream(modelFile) );
BaseClassifier compiledClassifier =
    (BaseClassifier) oi.readObject();
oi.close();

for(int i=0;i<CATEGORIES.length; i++ ) {
    Vector<FileBinayType> catVect = mainVector.get(i);
    String text;
    for(FileBinayType tf : catVect) {
        if(tf.state != null && tf.state == FileState.Test) {
            text = Files.readFromFile(tf.file,"UTF-8");
            ScoredClassification classification =
                (ScoredClassification)
                    compiledClassifier.classify(text);
            confMatrix.increment(CATEGORIES[i],
                classification.bestCategory());
        }
    }
} catch(Exception e) {
    e.printStackTrace();
}
return confMatrix;
}

////////////////////////////////////
////////////////////////////////////
//////////////////////////////////// Naive Bayes Classifier
////////////////////////////////////
////////////////////////////////////

public static void trainNaiveBayes(Vector<Vector> mainVector,
String modelFile) {
    NaiveBayesClassifier classifier =
        new NaiveBayesClassifier(CATEGORIES,
            new IndoEuropeanTokenizerFactory());

    try {
    for(int i=0;i<CATEGORIES.length; i++ ) {
        Vector<FileBinayType> catVect = mainVector.get(i);
        for(FileBinayType tf : catVect) {
            if(tf.state != null && tf.state == FileState.Train) {
                String text = Files.readFromFile(tf.file,"UTF-8");
                classifier.train(CATEGORIES[i],
                    text,text.length());
            }
        }
    }
    ObjectOutputStream os = new ObjectOutputStream(
        new FileOutputStream(modelFile));
    classifier.compileTo(os);
}

```

```

        os.close();
    }
} catch(IOException e) {
    e.printStackTrace();
}
}

////////////////////////////////
public static ConfusionMatrix testNaiveBayes(Vector<Vector>
mainVector, String modelFile) {
    ConfusionMatrix confMatrix =
        new ConfusionMatrix(CATEGORIES);
    try {
        ObjectInputStream oi = new ObjectInputStream(
            new FileInputStream(modelFile) );
        LMClassifier compiledClassifier =
            (LMClassifier) oi.readObject();
        oi.close();

        for(int i=0;i<CATEGORIES.length; i++ ) {
            Vector<FileBinayType> catVect = mainVector.get(i);
            String text;
            for(FileBinayType tf : catVect) {
                if(tf.state != null && tf.state == FileState.Test) {
                    text = Files.readFromFile(tf.file,"UTF-8");
                    JointClassification jc =
                        compiledClassifier.classifyJoint(
                            text.toCharArray(), 0, text.length());
                    confMatrix.increment(CATEGORIES[i],
                        jc.bestCategory());
                }
            }
        }
    } catch(Exception e) {
        e.printStackTrace();
    }
    return confMatrix;
}

////////////////////////////////
////////////////////////////////
////////////////////////////////
////////////////////////////////
////////////////////////////////
////////////////////////////////
////////////////////////////////
public static void trainTFIDF(Vector<Vector> mainVector, String
modelFile) {
    TfIdfClassifierTrainer<CharSequence> classifier= new
        TfIdfClassifierTrainer<CharSequence> (
            new TokenFeatureExtractor(
                new IndoEuropeanTokenizerFactory()) );
    try {
        for(int i=0;i<CATEGORIES.length; i++ ) {

```

```

Vector<FileBinayType> catVect = mainVector.get(i);

for(FileBinayType tf : catVect) {
    if(tf.state != null && tf.state == FileState.Train) {
        String text = Files.readFromFile(tf.file,"UTF-8");
        classifier.handle(new Classified(text,
            new Classification(CATEGORIES[i])));
    }
}
ObjectOutputStream os = new ObjectOutputStream(
    new FileOutputStream(modelFile));
classifier.compileTo(os);
os.close();
}
} catch(IOException e) {
    e.printStackTrace();
}
}

//////////
public static ConfusionMatrix testTFIDF(Vector<Vector> mainVector,
String modelFile) {
ConfusionMatrix confMatrix =
    new ConfusionMatrix(CATEGORIES);

try {
ObjectInputStream oi = new ObjectInputStream(
    new FileInputStream(modelFile) );
BaseClassifier compiledClassifier =
    (BaseClassifier) oi.readObject();
oi.close();

for(int i=0;i<CATEGORIES.length; i++ ) {
Vector<FileBinayType> catVect = mainVector.get(i);
String text;
for(FileBinayType tf : catVect) {
    if(tf.state != null && tf.state == FileState.Test) {
        text = Files.readFromFile(tf.file,"UTF-8");
        ScoredClassification classification =
            (ScoredClassification)
            compiledClassifier.classify(text);
        confMatrix.increment(CATEGORIES[i],
            classification.bestCategory());
    }
}
} catch(Exception e) {
    e.printStackTrace();
}
return confMatrix;
}

```



```

////////////////////////////////////
//////////
//////////////////////////////////// Utilities Function
////////////////////////////////////
//////////
public static void printConfusionMatrix(int[][]
tfidfConfMatricesAverage) {
    int numCats = CATEGORIES.length;

    System.out.print("      - ");
    for(int i=0;i<numCats;i++) {
        System.out.print(CATEGORIES[i] + " , ");
    }
    System.out.print("\n");

    for(int i=0;i<numCats;i++) {
        System.out.print(CATEGORIES[i] + " - ");
        for(int j=0;j<numCats;j++) {
            System.out.print(" " + tfidfConfMatricesAverage[i][j] +
" , ");
        }
        System.out.print("\n");
    }
}

public static int findMax(double... input) {
    Random rand = new Random();
    int index = rand.nextInt(input.length);

    double max = input[index];
    for(int i=0;i<input.length;i++) {
        if(max<input[i]) {
            max = input[i];
            index = i;
        }
    }
    return index;
}

public static double falsePositiveError(int[][] inMatrix) {
    int numL0inOther = 0;
    for(int i=1;i < CATEGORIES.length;i++) {
        numL0inOther += inMatrix[0][i];
    }

    int L0total = 0;
    for(int i=0;i < CATEGORIES.length;i++) {
        L0total += inMatrix[0][i];
    }

    int numL1inL3and4 = 0;
    for(int i=2;i < CATEGORIES.length;i++) {

```

```

        numL1inL3and4 += inMatrix[1][i];
    }

    int L1total = 0;
    for(int i=0;i < CATEGORIES.length;i++) {
        L1total += inMatrix[1][i];
    }

    return (double) (numL0inOther+numL1inL3and4) / (L0total+L1total);
}

public static int catStr2Int(String input) {
    for(int i=0;i<CATEGORIES.length;i++) {
        if(CATEGORIES[i].equals(input)) {
            return i;
        }
    }
    return -1;
}

public static double ScoreToProbability(ScoredClassification sc,
int rank) {
    double total = 0.0;
    for(int i=0;i<CATEGORIES.length;i++) {
        total += sc.score(i);
    }
    return (double) sc.score(rank) / total;
}

////////////////////////////////////
////////////////////////////////////
//////////////////////////////////// Main Function
////////////////////////////////////
////////////////////////////////////
public static void main(String[] args) {
    Vector<Vector> mainVector = null;
    Vector<ConfusionMatrix> tfidfConfMatrices =
        new Vector<ConfusionMatrix>();
    Vector<ConfusionMatrix> naivebayesConfMatrices =
        new Vector<ConfusionMatrix>();
    Vector<ConfusionMatrix> knnConfMatrices =
        new Vector<ConfusionMatrix>();
    Vector<ConfusionMatrix> lmConfMatrices =
        new Vector<ConfusionMatrix>();
    //combinations
    Vector<ConfusionMatrix> simpleVotingConfMatrices =
        new Vector<ConfusionMatrix>();
    Vector<ConfusionMatrix> dynamicClassifierSelctionConfMatrices =
        new Vector<ConfusionMatrix>();
    Vector<ConfusionMatrix>
adaptiveClassifierCombinationConfMatrices =
        new Vector<ConfusionMatrix>();
    Vector<ConfusionMatrix> newCombinationConfMatrices =

```

```

    new Vector<ConfusionMatrix>();

for(int run=1; run <= NUM_RUNS; run++) {
    //randomly select the files to be used for training and left
    // the remaining for testing
    mainVector = new Vector<Vector>();

    //Randomize Timer using Current Time of the system
    Calendar cal = Calendar.getInstance();
    Random randGen = new Random(cal.getTimeInMillis());

    try {
    int k = 0, cap = 0, length = 0;

    for(int i=0;i<CATEGORIES.length; i++ ) {
        Vector<FileBinayType> catVector =
            new Vector<FileBinayType>();
        File classDir = new File(ORIGINAL_DIR, CATEGORIES[i]);

        length = classDir.listFiles().length;
        cap = java.lang.Math.round( (float) length * PRCNT_TEST
);

        int[] testFilesIndex = new int[cap+1];
        for(int j=0; j <= cap; j++) {
            //Numbers should be unique
            int rn = randGen.nextInt(length);
            testFilesIndex[j] = rn;
            for(int p=0; p < j; p++) {
                if(testFilesIndex[p] == rn) {
                    j--;
                    break;
                }
            }
        }

        Arrays.sort(testFilesIndex);
        int s = 0;
        k = 0;
        for(File file : classDir.listFiles()) {
            FileBinayType tf;

            if(k <= cap && s==testFilesIndex[k]) {
                k++;
                tf = new FileBinayType(file,FileState.Test);
            } else {
                tf = new FileBinayType(file,FileState.Train);
            }
            catVector.add(tf);

            s++;
        }

        mainVector.add(catVector);
    }
}

```

```

    }
  } catch (Exception e) {
    e.printStackTrace();
  }

  //we will save the whole confusion matrix just in case for
  //future usage
  //all of the parameters that we need will be available
  //train the network using the following files

  System.out.println("===== "+run+
    " run =====");

  System.out.print("training ");

  trainTFIDF(mainVector, modelFileDir +
    "tcat_tfidfclassifier");
  System.out.print(" .");
  trainNaiveBayes(mainVector, modelFileDir +
    "tcat_naivebayesclassifier");
  System.out.print(" .");
  trainKnn(mainVector, modelFileDir + "tcat_knnclassifier",
2);
  System.out.print(" .");
  trainLM(mainVector, modelFileDir + "tcat_lmclassifier", 7);
  System.out.print(" .");
  trainNewClassifier(mainVector,
    modelFileDir + "new_lmclassifier",
    modelFileDir + "new_knnclassifier",
    modelFileDir + "new_naiveBayesclassifier",
    modelFileDir + "new_tfidfBayesclassifier",
    4, 7);

  System.out.println("... ");

  //test the network and see how good it is
  ConfusionMatrix confMatrix;

  confMatrix =
    testTFIDF(mainVector, modelFileDir +
    "tcat_tfidfclassifier");
    tfidfConfMatrices.add(confMatrix);

  System.out.println("TF/IDF Classifier ..... ");
  System.out.println(confMatrix.totalAccuracy());

  System.out.println("..... done");

  System.out.println(".....
  .....");

  confMatrix =

```

```

        testNaiveBayes(mainVector, modelFileDir +
            "tcat_naivebayesclassifier");
naivebayesConfMatrices.add(confMatrix);

System.out.println("Naive Bayes Classifier
..... ");
System.out.println(confMatrix.totalAccuracy());

System.out.println("..... done");

System.out.println(".....
.....");

confMatrix =
    testKnn(mainVector, modelFileDir +
"tcat_knnclassifier");
knnConfMatrices.add(confMatrix);

System.out.println("K Nearest Neighbour Classifier
..... ");
System.out.println(confMatrix.totalAccuracy());

System.out.println("..... done");

System.out.println(".....
.....");

confMatrix =
    testLM(mainVector, modelFileDir +
"tcat_lmclassifier");
lmConfMatrices.add(confMatrix);

System.out.println("Language Model Classifier
..... ");
System.out.println(confMatrix.totalAccuracy());

System.out.println("..... done");

System.out.println(".....
.....");

System.out.println("Combinations ..... ");
confMatrix =
    testSimpleVotingCombClassifier(mainVector,
        modelFileDir + "new_lmclassifier",
        modelFileDir + "new_knnclassifier",
        modelFileDir + "new_naiveBayesclassifier",
        modelFileDir + "new_tfidfBayesclassifier");
simpleVotingConfMatrices.add(confMatrix);
System.out.println("Simple Voting:"+

```

```

        confMatrix.totalAccuracy());

    confMatrix =
        testDynamicClassifierSelection(mainVector,
            modelFileDir + "new_lmclassifier",
            modelFileDir + "new_knnclassifier",
            modelFileDir + "new_naiveBayesclassifier",
            modelFileDir + "new_tfidfBayesclassifier");

    dynamicClassifierSelctionConfMatrices.add(confMatrix);
    System.out.println("Dynamic Classifier Selection: "+
        confMatrix.totalAccuracy());

    confMatrix =
        testAdaptiveClassifierComb(mainVector,
            modelFileDir + "new_lmclassifier",
            modelFileDir + "new_knnclassifier",
            modelFileDir + "new_naiveBayesclassifier",
            modelFileDir + "new_tfidfBayesclassifier");

    adaptiveClassifierCombinationConfMatrices.add(confMatrix);
    System.out.println("Adaptive Classifier Combination:"
        + confMatrix.totalAccuracy());

    confMatrix =
        testNewCombClassifier(mainVector,
            modelFileDir + "new_lmclassifier",
            modelFileDir + "new_knnclassifier",
            modelFileDir + "new_naiveBayesclassifier",
            modelFileDir + "new_tfidfBayesclassifier");
    newCombinationConfMatrices.add(confMatrix);
    System.out.println("New Classifier Combination: "+
        confMatrix.totalAccuracy());

    System.out.println("..... done");

    System.out.println("=====");
    =="+
    "=====\n\n";
    }

    System.out.println("Calculating Results . . .");
    //get average the results and print the final results
    int numCats = CATEGORIES.length;

    int[][] tfidfConfMatricesAverage = new int[numCats][numCats];
    int[][] naivebayesConfMatricesAverage = new
        int[numCats][numCats];
    int[][] knnConfMatricesAverage = new int[numCats][numCats];
    int[][] lmConfMatricesAverage = new int[numCats][numCats];

```

```

int[][] simpleVotingConfMatricesAverage = new
    int[numCats][numCats];
int[][] dcsConfMatricesAverage = new int[numCats][numCats];
int[][] accConfMatricesAverage = new int[numCats][numCats];
int[][] newConfMatricesAverage = new int[numCats][numCats];

for(int i=0;i<numCats;i++) {
    for(int j=0;j<numCats;j++) {
        tfidfConfMatricesAverage[i][j] = 0;
        naivebayesConfMatricesAverage[i][j] = 0;
        knnConfMatricesAverage[i][j] = 0;
        lmConfMatricesAverage[i][j] = 0;
        simpleVotingConfMatricesAverage[i][j] = 0;
        dcsConfMatricesAverage[i][j] = 0;
        accConfMatricesAverage[i][j] = 0;
        newConfMatricesAverage[i][j] = 0;
    }
}

for(int i=0;i<numCats;i++) {
    for(int j=0;j<numCats;j++) {
        for(int k=0;k<NUM_RUNS;k++) {
            tfidfConfMatricesAverage[i][j] +=
                tfidfConfMatrices.get(k).matrix()[i][j];

            naivebayesConfMatricesAverage[i][j] +=
                naivebayesConfMatrices.get(k).matrix()[i][j];

            knnConfMatricesAverage[i][j] +=
                knnConfMatrices.get(k).matrix()[i][j];

            lmConfMatricesAverage[i][j] +=
                lmConfMatrices.get(k).matrix()[i][j];

            simpleVotingConfMatricesAverage[i][j] +=
                simpleVotingConfMatrices.get(k).matrix()[i][j];

            dcsConfMatricesAverage[i][j] +=
                dynamicClassifierSelctionConfMatrices.
                get(k).matrix()[i][j];

            accConfMatricesAverage[i][j] +=
                adaptiveClassifierCombinationConfMatrices.
                get(k).matrix()[i][j];

            newConfMatricesAverage[i][j] +=
                newCombinationConfMatrices.get(k).matrix()[i][j];
        }

        tfidfConfMatricesAverage[i][j] /= (double)NUM_RUNS;
        naivebayesConfMatricesAverage[i][j] /= (double)NUM_RUNS;
        knnConfMatricesAverage[i][j] /= (double)NUM_RUNS;
        lmConfMatricesAverage[i][j] /= (double)NUM_RUNS;
    }
}

```

```

    simpleVotingConfMatricesAverage[i][j] /= (double) NUM_RUNS;
    dcsConfMatricesAverage[i][j] /= (double) NUM_RUNS;
    accConfMatricesAverage[i][j] /= (double) NUM_RUNS;
    newConfMatricesAverage[i][j] /= (double) NUM_RUNS;
}
}
////////////////////////////////////
//calculate total accuracy average
double tfidfTotalAccuracyAverage = 0.0;
double naivebayesTotalAccuracyAverage = 0.0;
double knnTotalAccuracyAverage = 0.0;
double lmTotalAccuracyAverage = 0.0;
double simpleVotingTotalAccuracyAverage = 0.0;
double dcsTotalAccuracyAverage = 0.0;
double accTotalAccuracyAverage = 0.0;
double newTotalAccuracyAverage = 0.0;

for(int i=0;i<NUM_RUNS;i++) {
    tfidfTotalAccuracyAverage +=
        tfidfConfMatrices.get(i).totalAccuracy();

    naivebayesTotalAccuracyAverage +=
        naivebayesConfMatrices.get(i).totalAccuracy();

    knnTotalAccuracyAverage +=
        knnConfMatrices.get(i).totalAccuracy();

    lmTotalAccuracyAverage +=
        lmConfMatrices.get(i).totalAccuracy();

    simpleVotingTotalAccuracyAverage +=
        simpleVotingConfMatrices.get(i).totalAccuracy();

    dcsTotalAccuracyAverage +=
        dynamicClassifierSelctionConfMatrices.get(i).
            totalAccuracy();

    accTotalAccuracyAverage +=
        adaptiveClassifierCombinationConfMatrices.get(i).
            totalAccuracy();

    newTotalAccuracyAverage +=
        newCombinationConfMatrices.get(i).
            totalAccuracy();
}

tfidfTotalAccuracyAverage /= (double) NUM_RUNS;
naivebayesTotalAccuracyAverage /= (double) NUM_RUNS;
knnTotalAccuracyAverage /= (double) NUM_RUNS;
lmTotalAccuracyAverage /= (double) NUM_RUNS;
simpleVotingTotalAccuracyAverage /= (double) NUM_RUNS;
dcsTotalAccuracyAverage /= (double) NUM_RUNS;
accTotalAccuracyAverage /= (double) NUM_RUNS;

```



```

newTotalAccuracyAverage /= (double) NUM_RUNS;
////////////////////////////////////
//calculate total type I and II errors
double lmTotalFalsePositive =
    falsePositiveError(lmConfMatricesAverage);
double tfidfTotalFalsePositive =
    falsePositiveError(tfidfConfMatricesAverage);
double knnTotalFalsePositive =
    falsePositiveError(knnConfMatricesAverage);
double naivebayesTotalFalsePositive =
    falsePositiveError(naivebayesConfMatricesAverage);
double simpleVotingTotalFalsePositive =
    falsePositiveError(simpleVotingConfMatricesAverage);
double dcsTotalFalsePositive =
    falsePositiveError(dcsConfMatricesAverage);
double accTotalFalsePositive =
    falsePositiveError(accConfMatricesAverage);
double newTotalFalsePositive =
    falsePositiveError(newConfMatricesAverage);
////////////////////////////////////
/// Recall
double tfidfRecallAverage = 0.0;
double naivebayesRecallAverage = 0.0;
double knnRecallAverage = 0.0;
double lmRecallAverage = 0.0;
double simpleVotingRecallAverage = 0.0;
double dcsRecallAverage = 0.0;
double accRecallAverage = 0.0;
double newRecallAverage = 0.0;

for(int i=0;i<NUM_RUNS;i++) {
    tfidfRecallAverage +=
        tfidfConfMatrices.get(i).macroAvgRecall();

    naivebayesRecallAverage +=
        naivebayesConfMatrices.get(i).macroAvgRecall();

    knnRecallAverage +=
        knnConfMatrices.get(i).macroAvgRecall();

    lmRecallAverage +=
        lmConfMatrices.get(i).macroAvgRecall();

    simpleVotingRecallAverage +=
        simpleVotingConfMatrices.get(i).macroAvgRecall();

    dcsRecallAverage +=
        dynamicClassifierSelctionConfMatrices.get(i).
        macroAvgRecall();

    accRecallAverage +=
        adaptiveClassifierCombinationConfMatrices.get(i).
        macroAvgRecall();
}

```

```

newRecallAverage +=
    newCombinationConfMatrices.get(i).
        macroAvgRecall();
}

tfidfRecallAverage /= (double) NUM_RUNS;
naivebayesRecallAverage /= (double) NUM_RUNS;
knnRecallAverage /= (double) NUM_RUNS;
lmRecallAverage /= (double) NUM_RUNS;
simpleVotingRecallAverage /= (double) NUM_RUNS;
dcsRecallAverage /= (double) NUM_RUNS;
accRecallAverage /= (double) NUM_RUNS;
newRecallAverage /= (double) NUM_RUNS;

////////////////////////////////////
////////////////////////////////////
//print the results
System.out.println("TF-IDF Confusion Matrix\n");
printConfusionMatrix(tfidfConfMatricesAverage);
System.out.println("\nAverage of Total Accuracy = "+
    tfidfTotalAccuracyAverage);
System.out.println("\nTotal Error = "+
    tfidfTotalFalsePositive);
System.out.println("\nRecall = "+
    tfidfRecallAverage);

System.out.println("Naive Bayes Confusion Matrix\n");
printConfusionMatrix(naivebayesConfMatricesAverage);
System.out.println("\nAverage of Total Accuracy = "+
    naivebayesTotalAccuracyAverage);
System.out.println("\nTotal Error = "+
    naivebayesTotalFalsePositive);
System.out.println("\nRecall = "+
    naivebayesRecallAverage);

System.out.println("Knn Confusion Matrix\n");
printConfusionMatrix(knnConfMatricesAverage);
System.out.println("\nAverage of Total Accuracy = "+
    knnTotalAccuracyAverage);
System.out.println("\nTotal Error = "+
    knnTotalFalsePositive);
System.out.println("\nRecall = "+
    knnRecallAverage);

System.out.println("LM Confusion Matrix\n");
printConfusionMatrix(lmConfMatricesAverage);
System.out.println("\nAverage of Total Accuracy = "+
    lmTotalAccuracyAverage);
System.out.println("\nTotal Error = "+

```

```

        lmTotalFalsePositive);
System.out.println("\nRecall = "+
        lmRecallAverage);

System.out.println("=====  

=====");

System.out.println("Simple Voting Confusion Matrix\n");

printConfusionMatrix(simpleVotingConfMatricesAverage);
System.out.println("\nAverage of Total Accuracy = "+
        simpleVotingTotalAccuracyAverage);
System.out.println("\nTotal Error = "+
        simpleVotingTotalFalsePositive);
System.out.println("\nRecall = "+
        simpleVotingRecallAverage);

System.out.println("Dynamic Classifier Selection Confusion  

Matrix\n");
printConfusionMatrix(dcsConfMatricesAverage);
System.out.println("\nAverage of Total Accuracy = "+
        dcsTotalAccuracyAverage);
System.out.println("\nTotal Error = "+
        dcsTotalFalsePositive);
System.out.println("\nRecall = "+
        dcsRecallAverage);

System.out.println("Adaptive Classifier Combination  

Confusion Matrix\n");
printConfusionMatrix(accConfMatricesAverage);
System.out.println("\nAverage of Total Accuracy = "+
        accTotalAccuracyAverage);
System.out.println("\nTotal Error = "+
        accTotalFalsePositive);
System.out.println("\nRecall = "+
        accRecallAverage);

System.out.println("NEW Classifier Combination Confusion  

Matrix\n");
printConfusionMatrix(newConfMatricesAverage);
System.out.println("\nAverage of Total Accuracy = "+
        newTotalAccuracyAverage);
System.out.println("\nTotal Error = "+
        newTotalFalsePositive);
System.out.println("\nRecall = "+
        newRecallAverage);

}

```

}

It is worth mentioning here that this work is an implementation based on LingPipe open source library so it is also open source and free, having the same license as LingPipe <http://alias-i.com/lingpipe/licenses/lingpipe-license-1.txt>.

Appendix VI

The sample output of program for a single run:

```

===== 1 run
=====
training .....
TF/IDF Classifier .....
0.6666666666666666
..... done
.....
Naive Bayes Classifier .....
0.7333333333333333
..... done
.....
K Nearest Neighbour Classifier .....
0.5333333333333333
..... done
.....
Language Model Classifier .....
0.7666666666666667
..... done
.....
Combinations .....
Simple Voting: 0.7666666666666667
Dynamic Classifier Selection: 0.7
Adaptive Classifier Combination: 0.7666666666666667
New Classifier Combination: 0.8
..... done
=====
=====

```

Calculating Results . . .

TF-IDF Confusion Matrix

```

- level0 , level1 , level2 , level3 ,
level0 - 3 , 2 , 0 , 0 ,
level1 - 0 , 4 , 0 , 1 ,
level2 - 0 , 0 , 1 , 4 ,
level3 - 0 , 2 , 1 , 12 ,

```

Average of Total Accuracy = 0.6666666666666666

Total Error = 0.3

Recall = 0.6

Naive Bayes Confusion Matrix

```

- level0 , level1 , level2 , level3 ,
level0 - 4 , 1 , 0 , 0 ,

```

```
level1 - 1 , 3 , 0 , 1 ,
level2 - 0 , 0 , 1 , 4 ,
level3 - 0 , 1 , 0 , 14 ,
```

Average of Total Accuracy = 0.7333333333333333 ...

Total Error = 0.2

Recall = 0.6333333333333333
Knn Confusion Matrix

```
      - level0 , level1 , level2 , level3 ,
level0 - 2 , 3 , 0 , 0 ,
level1 - 1 , 3 , 0 , 1 ,
level2 - 0 , 1 , 2 , 2 ,
level3 - 2 , 2 , 2 , 9 ,
```

Average of Total Accuracy = 0.5333333333333333

Total Error = 0.4

Recall = 0.5
LM Confusion Matrix

```
      - level0 , level1 , level2 , level3 ,
level0 - 4 , 1 , 0 , 0 ,
level1 - 1 , 3 , 0 , 1 ,
level2 - 0 , 0 , 1 , 4 ,
level3 - 0 , 0 , 0 , 15 ,
```

Average of Total Accuracy = 0.7666666666666667

Total Error = 0.2

Recall = 0.6499999999999999
=====
Simple Voting Confusion Matrix

```
      - level0 , level1 , level2 , level3 ,
level0 - 5 , 0 , 0 , 0 ,
level1 - 1 , 3 , 0 , 1 ,
level2 - 0 , 0 , 1 , 4 ,
level3 - 0 , 1 , 0 , 14 ,
```

Average of Total Accuracy = 0.7666666666666667

Total Error = 0.1

Recall = 0.6833333333333333
Dynamic Classifier Selection Confusion Matrix

```
      - level0 , level1 , level2 , level3 ,
level0 - 2 , 3 , 0 , 0 ,
```

```
level1 - 1 , 3 , 0 , 1 ,
level2 - 0 , 0 , 1 , 4 ,
level3 - 0 , 0 , 0 , 15 ,
```

Average of Total Accuracy = 0.7

Total Error = 0.4

Recall = 0.55

Adaptive Classifier Combination Confusion Matrix

```
      - level0 , level1 , level2 , level3 ,
level0 - 5 , 0 , 0 , 0 ,
level1 - 1 , 4 , 0 , 0 ,
level2 - 0 , 0 , 0 , 5 ,
level3 - 0 , 1 , 0 , 14 ,
```

Average of Total Accuracy = 0.7666666666666667

Total Error = 0.0

Recall = 0.6833333333333333

NEW Classifier Combination Confusion Matrix

```
      - level0 , level1 , level2 , level3 ,
level0 - 5 , 0 , 0 , 0 ,
level1 - 1 , 4 , 0 , 0 ,
level2 - 0 , 0 , 2 , 3 ,
level3 - 0 , 1 , 1 , 13 ,
```

Average of Total Accuracy = 0.8

Total Error = 0.0

Recall = 0.7666666666666667

