RECORDS ANALYSIS AND CLASSIFICATION SYSTEM: A PROOF OF CONCEPT SYSTEM FOR THE AUTOMATED CLASSIFICATION OF UNITED STATES AIR FORCE RECORDS

THESIS

David W. Snoddy, Captain, USAF

AFIT/GIR/LAR/96D-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

AFIT/GIR/LAR/96D-11

RECORDS ANALYSIS AND CLASSIFICATION SYSTEM: A PROOF OF CONCEPT SYSTEM FOR THE AUTOMATED CLASSIFICATION OF UNITED STATES AIR FORCE RECORDS

THESIS

Presented to the Faculty of the Graduate School of Logistics

and Acquisition Management of the

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Information Resources Management

David W. Snoddy, B.A.

Captain, USAF

December 1996

Acknowledgments

This thesis was a massive undertaking which could not have been completed without the help and encouragement of others. First, I would like to thank Dr. Kim Campbell, my advisor, and Dr. Guy Shane, my reader. Their guidance and expertise made the entire process a lot less painful than it could have been. Next, I would like to thank Capt Anne McPharlin, my sponsor, and the personnel from Wright-Patterson's Base Records Management office for providing various reference materials for my use throughout this process. I would also like to extend my thanks to the personnel in the 88th Support Group Administration Office for allowing me to rifle through their official files in order to accumulate my sample of records. Most of all, I would like to thank the most important person, my wife. Without her understanding and support this thesis would not have been possible.

David W. Snoddy

Table of Contents

Page

Page

List of Figures

Figure Page

Figure Page

Figure Page

Figure Page

List of Tables

Abstract

The records management process utilized within the Department of Defense (DoD) is currently labor intensive. Work is being done to automate portions of this process, but classifying documents and assigning disposition instructions remains a labor intensive, manual operation. Although the requirement for this capability was identified by a DoD sponsored study, an automated computer-based system which can classify and apply disposition instructions has yet to be developed for use within the DoD. This thesis study presents a proof of concept computer program called the Records Analysis and Classification System (RACS) which was developed to demonstrate computer-based techniques for the automated classification of official records. To demonstrate the operation of RACS, a sample of 113 records was collected from the files of an organization at Wright-Patterson AFB. An analysis of the results of the tests conducted with the RACS system indicated that it was capable of accurately classifying 72 out of the 113 records on average. Additionally, the RACS program was designed as a learning system and the test results indicated that it was, in fact, capable of improving its classification accuracy over time.

RECORDS ANALYSIS AND CLASSIFICATION SYSTEM: A PROOF OF CONCEPT SYSTEM FOR THE AUTOMATED CLASSIFICATION OF UNITED STATES AIR FORCE RECORDS

I. Introduction

In recent years there has been an increasing awareness of a need to manage our military's information resources more effectively. The Department of Defense (DoD), recognizing this fact, has initiated some programs to address various areas of concern. One such area has been the process of records management. To illustrate the need for a fresh approach the following section provides some details on the current process in place in the United States Air Force (USAF). To document some of the work done by the DoD in order to overcome the limitations of the current records management process, the second section presents a chronological list of such activities.

United States Air Force Records Management

The United States Air Force uses, Air Force Instruction 37-122 Air Force Records Management Program (AFI 37-122), Air Force Manual 37-123 Management of Records (AFMAN 37-123), and Air Force Manual 37-139 Records Disposition Schedule (AFMAN 37-139, formerly AFR 4-20 V2) to manage its official records. AFMAN 37-139 is used specifically to manage the classification and disposition of official records (SECAF, 1996:1). The disposition instructions for a given record can be found in one of

438 decision logic tables (DLT) contained in AFMAN 37-139. Related DLTs are grouped under one of 39 different series. For example, DLTs dealing with the area of Information Management are in series 37 while those DLTs dealing with Acquisition are in series 63 (note that the series are not numbered consecutively, their numbers correspond to the appropriate governing instruction series) (SECAF, 1996:1). In all, there are approximately 6150 disposition rules prescribed in AFMAN 37-139.

The disposition of records is managed by Records Technicians, primarily clerks and secretaries, under the direction of a Chief of an Office of Record (SECAF, 1994a:Sec 8-9). Records Technicians and Chiefs of an Office of Record are assisted by a Functional Area Records Manger who is advised by the base Records Manager (SECAF, 1994a:Sec 6-9). Records Technicians develop files maintenance and disposition plans (files plans) and physically file and manage (primarily) paper records. As the Air Force draws down, the first slots to be eliminated are often clerks and secretaries, placing such administrative tasks on the other unit personnel (McPharlin, 1995). Additionally, desktop PCs and local area networks are providing the capability to create, use, maintain and disseminate records electronically. The majority of these electronic records (including e-mail) are not being managed by the unit files plans (McPharlin, 1995; Bolden and Pollard, 1996). This results in lost information and/or information retained beyond its disposition, which takes up valuable disk space and could leave the unit vulnerable to Freedom of Information Act (FOIA) requests or lawsuits (McPharlin, 1995; Bolden and Pollard, 1996). These records are not managed under the current process because an understanding of the myriad tables

and rules and knowledge of public law is a burden upon the typical end-user or producer of USAF records.

Records Management and the DoD

In July 1994 the Department of Defense Records Management Business Process Reengineering (RM-BPR) study was completed. The study was sponsored by the Assistant Secretary of Defense for Command, Control, Communications and Intelligence (ASD(C^3I)), and the Deputy Assistant Secretary of Defense for Information Management (DASD(IM)). During the course of the study representatives from each military service and the Office of the Secretary of Defense reengineered the process of records management (DoD RM-BPR, 1994:v).

In September 1994 the ASD(C^3I) directed the DASD(IM) to create the Department of Defense Records Management Task Force (RMTF). The designated mission of the RMTF is to develop plans and draft policy to implement, by the year 2003, the initiatives proposed by the RM-BPR (DoD RMTF, 1995: ES-1).

The RM-BPR identified six opportunities for improving records management within the DoD. These six opportunities became the six strategic policy initiatives for the RMTF (DoD RMTF, 1995:ES-1). The initiative of interest in this thesis is, "Develop standard DoD functional and automated systems requirements for managing information as records in an electronic environment" (DoD RMTF, 1995:ES-1). The DoD has released a draft which "sets forth mandatory baseline functional requirements and data elements for storing and accessing information from Records Management Application (RMA) software used by DoD agencies in the implementation of their records

management programs" (DoD, 1996:1). However, this draft standard does not address one of the original functional support requirements proposed by the RM-BPR which was to, "Assign disposition instructions automatically" (DoD RM-BPR, 1994:4-2). It is this original requirement which is of interest in this thesis.

Problem Statement

The records management process is currently very labor-intensive. Work is being done to automate portions of this process but the process of classifying documents and assigning disposition instructions remains a labor-intensive, manual operation. Although the requirement for this capability was identified by the RM-BPR study, an automated computer-based system which could accomplish the classification and disposition assignment process has yet to be developed. (Note: throughout this document the term *disposition* is synonymous with *disposition instructions*)

Research Objectives

The following objectives must be met in order to solve the specific problem of interest:

1. Locate and summarize the various automatic document classification techniques being employed by researchers and practitioners on related projects throughout the world.
2. Develop and propose a technique for automatically analyzing records in order to assign appropriate classification and disposition within the USAF.
3. Demonstrate the proposed technique on a limited set of sample records.

Scope and Limitations of the Study

The process of computer-based records management is a much larger process than simply classifying records and assigning disposition instructions. The draft DoD Standard 5015.2 specifies 13 broad functions a RMA should be capable of performing; some examples include: Identifying Records, Filing Records and Assigning Disposition, Storing Records, Retrieving Records, and Destruction of Records (DoD, 1996:14-21). This thesis effort is not concerned with the whole process; instead, the scope of this project is limited to the portion of the process which currently requires a human records technician to determine record type and assign appropriate classification and disposition instructions based on that determination.

Recognizing the fact that the DoD is currently engaged in an effort to standardize and simplify the records management process across the DoD, this study will not duplicate that effort. In other words, this study will not make any attempt to revise and/or propose a new records schedule; rather it will demonstrate a technique which can be applied regardless of the underlying schedule.

Summary

As is evident from the information presented thus far, the process of classifying USAF records is not a simple process. The RM-BPR study recognized that one way to improve the efficiency of the records management process is to have disposition instructions assigned automatically (DoD RM-BPR, 1994:4-2). DoD-STD-5015.2 omitted this requirement (DoD:1996); consequently, any RMA software developed will undoubtedly require the user to choose the appropriate disposition manually. This thesis

has proposed to develop and demonstrate a technique for automatic classification which will fill in the gap left by DoD-STD-5015.2.

Chapter II, the literature review, provides general background on some relevant artificial intelligence technologies and on document classification projects from the literature.

II. Literature Review

Introduction

This literature review is subdivided into several different sections. The first several sections are designed to provide a background on artificial intelligence (AI) and on two specific AI areas of interest in this research effort. The remaining sections detail specific research which has been conducted previously in the area of applying AI techniques to the classification of documents.

Artificial Intelligence

Morris Firebaugh presents the following definition of AI which he attributes to Professor Marvin Minsky of MIT, "Artificial intelligence is the science of making machines do things that would require intelligence if done by men" (1988:12). This is not the only definition of AI; in fact, definitions abound. The underlying principle remains the same: artificial intelligence is a general term applied to the systems and techniques, usually computerized, which are capable of performing functions which are normally considered to require intelligence. Two examples include natural language processing and visual perception. The two specific AI areas which are addressed within this thesis are knowledge-based systems and natural language processing. These two areas are summarized here due to their applicability to automatic document classification tasks.

Knowledge-Based Systems

A knowledge-based system can be defined as "a computer system that attempts to replicate specific human expert intelligent activities" (Mockler and Dologite, 1992:14). One specific, well-known form of a knowledge-based system is the expert system. Expert systems have been developed for a variety of diverse tasks ranging from medical diagnosis programs such as MYCIN to speech understanding programs such as HEARSAY-II (Mockler and Dologite, 1992:17).

There are several key components which distinguish a knowledge-based system from other computer-based systems: a knowledge base, inference mechanism, user interface, and working memory (Firebaugh, 1988:337-338; Mockler and Dologite, 1992:19-21; Goel, 1994:54). The knowledge base contains domain-specific information and heuristics which pertain to the domain of interest. For example, the knowledge base of the MYCIN system contained about 400 heuristic IF-THEN rules pertaining to the diagnosis and treatment of infectious blood diseases (Hayes-Roth, 1992:16). The knowledge base can be considered a codified version of the knowledge derived from the experts within the particular domain. The inference mechanism is the component in a knowledge-based system which matches the input supplied against the knowledge contained in the knowledge base (Goel, 1994:54). For instance, the MYCIN system prompts the user with a series of questions. Depending on the answers to various questions, between 30 and 90 questions may be asked before a diagnosis is reached (Firebaugh, 1988:352-353). Once MYCIN has gathered sufficient information, it produces a diagnosis which includes the

most likely causes of infection as well as the recommended treatment (Firebaugh, 1988:353).

There are a variety of ways knowledge can be represented in a knowledge base. The most common way to store knowledge is in the form of rules, often called production rules. A typical rule uses if-then statements and Boolean operators to assign values to variables based on an analysis of input data. An example from the MYCIN system is illustrated in Figure 1 (Firebaugh, 1988:309).

IF	[a) the stain of the organism is gramneg AND b) the morphology of the organism is rod AND c) the patient is a compromised host]
THEN	[there is suggestive evidence (.6) that the identity of the organism is pseudamonas]

Figure 1. Example of IF-THEN Rules

Another method of storing knowledge is in frames. The underlying concept of frames is to store pieces of knowledge together in meaningful chunks; for example, a frame for a book might contain the title, author, publisher, and date of publication (Weckert, 1992:31).

Natural Language Processing

In very general terms, a natural language processing (NLP) system can be defined as "any system which performs a useful operation on natural language input" (Firebaugh, 1988:237). NLP systems have been developed for a number of diverse purposes ranging from lexical and syntactic analysis tools such as spelling checkers to much more intensive

applications such as those for speech recognition (Firebaugh, 1988:239). This research effort is concerned with the more general NLP processes for text analysis and will focus the discussion on those areas.

Statistical text analysis is employed frequently for automatic classification tasks (see Cheng and Wu, 1995; Losee and Haas, 1995; Larson, 1992). Statistical text analysis is often considered to have as its origin the early works of Luhn. Luhn, as quoted in Van Rijsbergen, stated, "It is here proposed that the frequency of word occurrence in an article furnishes a useful measurement of word significance" (1979:15). It is on this simple premise that much of the subsequent work in automatic term indexing has been built. An in-depth discussion of Luhn's work is not appropriate here but a summary of his technique can be found in Van Rijsbergen (1979:15-16).

If one were to produce a list of all like words with their frequencies from a given set of documents, it is not hard to imagine that the word list would quickly become quite long. Thanks to the work of Luhn and other subsequent researchers there are techniques for reducing these word lists to a more manageable length which will yield a list of index terms or keywords which will be representative of the original document. The basic steps employed typically are:

1. Removal of high frequency words or *stopwords*. This can be accomplished by means of a *stoplist* which is a list of all those words which occur frequently and are not considered key words (Van Rijsbergen 1979:17). Examples of stopwords are such function words as: on, the, about, has, now, and which.

2. Stripping of suffixes. This process, also called conflation or stemming, involves examining words for particular word-endings and removing them (Paice, 1977: 82). Examples include such word endings as -ing and -ous.

3. Detection of equivalent words. For example, MATHMATICS should be reduced to MATH (Cheng and Wu, 1995:294).

The discussion above is intended to provide a very simplified, brief overview of the techniques involved in automatic text analysis. As will be seen in the following discussion, refinements and adaptations of the basic text analysis process are integral portions of the various systems discussed.

Document Classification Systems

It is appropriate at this point to define some key terms used frequently throughout this thesis. The term *document* is used generically to refer to any form of written material to be classified whether it be a book, journal article, or an office memo. *A document classification system* is defined as a system which takes as its input a document to be classified and produces as its output the correct classification for the given document in relation to a predefined classification scheme.

Considering the definition presented above, it can be said that there are two key processes which every document classification system must perform. The first process is *text analysis*. The second is *determination of document classification* or *classification determination* for short. Taken together, the output of *text analysis* is a *document representation* which can be utilized by the system in order to determine the appropriate classification during *classification determination.*

It has been noted by various authors and confirmed by this author that much of the research done in the 1970s and '80s with automatic classification focused on clustering like documents without regard to any predefined classification scheme (Larson 1992:131;

Cheng and Wu, 1995:289). Given the focus of this thesis effort and the definitions presented above, the following discussion does not address this early research; instead it focuses on a sample of contemporary research.

For clarity of discussion, the various classification projects surveyed below are broken into three sections. The first section--Manual, Knowledge-Based Systems--describes two knowledge-based systems which require the user to input the significant data to the system in order for it to make a classification. The second section--Automated, Knowledge-Based Systems--describes one project which incorporates automated document content analysis with a knowledge-based representation of the classification scheme. The third section--Automated, Mathematical Comparison Systems--describes three systems which were each designed to conduct autonomous analysis of document content and arrive at a classification using various mathematical techniques.

Manual Knowledge-Based Systems. A perfect example of a manual/user-dependent document classification system is the current USAF document classification system. An example of a classification system which basically automates a portion of what still remains a largely manual system is the CLOD-X expert system. CLOD-X, which is the acronym for Classification of Office Document Expert System, was designed for records managers in the International Civil Aviation Organization (Savic, 1994:20). The system's knowledge base utilizes both rules and frames to represent the knowledge necessary for classification. Classification is accomplished through a process which requires the user to analyze a given document and then answer a series of questions posed

by CLOD-X. Once CLOD-X has gathered sufficient information from the user, it returns a document classification from a possible 400 classes.

Cosgrave and Weimann present a discussion on the use of an expert system tool known as n-Cube for item classification using the Universal Decimal Classification (UDC) standard (1992:33). The system they describe requires the user to make some preliminary assessments of the documents, basically determining keywords and concepts associated with the document. The user feeds these keywords into the expert system, and the system returns a suggestion as to the appropriate UDC classification number. While this system does not require the user to answer a series of questions as with the CLOD-X system, it still requires the user to analyze the source document and input an accurate set of keywords to receive a document classification from the system.

Automated, Knowledge-Based Systems. The one system which falls into this category is the one described by Bhatia et al. The bulk of their article describes the process of creating a knowledge base for their classification system. To construct the knowledge base the authors borrow a concept from the field of clinical psychology known as Personal Construct Theory (Bhatia et al., 1991:92). The system developers then apply the techniques of this theory during the process of knowledge elicitation conducted with classification experts. The resulting knowledge base is composed of a series of production rules.

A new document is automatically analyzed to extract index terms or term phrases. The authors refer to these terms or term phrases as constructs (Bhatia et al., 1991:96). A given document is therefore represented by a set of constructs. The occurrence of a

construct in the document triggers the rules corresponding to that construct in the system's knowledge base. The document is ultimately assigned the classification of the category for which the calculated certainty of correct classification is greatest (Bhatia et al., 1991:96).

This article is of value in that it illustrates one method for developing a knowledge base for the classification of documents. The one significant drawback of their system is that it is extremely labor intensive during the development phase. The systems presented in the next section illustrate some methods for automating this phase of system development as well.

Automated, Mathematical Comparison Systems. The projects and techniques presented in the previous two sections each used expert system techniques to represent the knowledge of a particular classification scheme. In contrast, the three projects presented in this section depart from this approach and instead use various statistical comparison approaches to determine document classification.

The first project reviewed (Losse and Haas, 1995) had several stages. First the researchers looked at word frequencies and especially at the frequencies of what they called sublanguage terms. The authors define a sublanguage as "the written or spoken language that is used in a particular field or discipline by people working in the field, especially to communicate with their colleagues" (Losee and Haas, 1995:519). The study focused on eight general fields or disciplines: biology, economics, electrical engineering, history, math, physics, psychology, and sociology.

During the second phase of their project, they conducted an investigation of how many terms they found to be sublanguage terms in the titles and abstracts of various articles as defined by special dictionaries for each discipline and by a general dictionary.

The final portion of their investigation is the one of most interest in this thesis. The authors developed a system to test whether term frequencies could be used to accurately classify a document--represented in their study by the abstract within a document--into its correct discipline.

Each discipline was represented in the system by a database of its sublanguage terms along with their Poisson percentiles. The Poisson percentile as presented in this study, "provides a measure of the degree to which a term has a higher than expected frequency of occurrence in the database in question" (Losee and Haas, 1995:522). To determine the correct discipline of a given abstract, the list of words in that abstract would be compared with each of the eight lists of words in the discipline databases. Poisson percentiles were calculated for the abstract in relation to each database and a composite score or weight computed (Losee and Haas, 1995:527). The abstract was then classified as a member of one of the eight disciplines based on the highest composite score or weight (Losee and Haas, 1995:527).

Between 22 and 50 abstracts from each discipline were presented to the system to determine its ability to accurately characterize the general domain to which the abstract belonged. The results of the experiment yielded amazing results. The authors' system was highly accurate, with the lowest success rate at 92.3% while classifying documents from the general domain, history (Losee and Haas, 1995:527). The system was 100%

accurate in classifying both math and sociology documents (Losee and Haas, 1995:527). The higher error rates for the history domain were attributed to the fact that the sub-language for that domain did not contain as many unique terms as the sub-language for an area like mathematics (Losee and Haas, 1995:528).

The experiment developed and reported by Larson attempted to select the correct classification for a document based on the characteristics of that document and on the characteristics of all documents previously assigned the same classification. The classification categories used in the study were from the Library of Congress Classification scheme. Larson developed what he called *classification clusters* to represent each Library of Congress Classification tested in the study. The classification clusters were essentially weighted vectors of index terms (Larson, 1992:132).

Larson's basic classification technique was to use the terms extracted from a document to be classified as a query to a set of databases where each database represented a classification cluster. The results of these queries indicated the document's classification. Larson conducted an exhaustive set of experiments using each combination of four matching methods, five query types and three index term representation schemes.

Larson reported that the highest accuracy achieved by any single combination of the parameters specified above yielded an accuracy of 46.6% (Larson, 1992:145). Larson was less than optimistic in his conclusion as to the effectiveness of a fully autonomous classification system. He stated that "fully automatic LC (Library of Congress) classification may not be possible for all books. A semiautomatic method of

classification, using one of a combination of the methods tested here, followed by human examination and selection of the highly ranked clusters, appears to be feasible" (Larson, 1992:146).

The last article reviewed is a study by Cheng and Wu which investigated the feasibility of automatic classification under the Universal Decimal Classification (UDC) scheme. Their technique was very similar to the others presented within this section. First class vectors were developed for each class used in the study. These class vectors essentially consisted of the keyterms and their frequencies from a set of sample books. When a new book was to be classified, a book vector would be generated. The book's title and chapter headings were used to generate the keyterms which made up each book vector. Once a book vector was developed it was compared to each of the class vectors and, using a calculation called the Modified Overlap Coefficient, a measure of similarity was determined (Cheng and Wu, 1995:293). The class vector which yielded the highest similarity was the one to which the new book was assigned. The results using this technique appear very promising. The authors report that when 384 books were classified about 86% were classified correctly (Cheng and Wu, 1995:296).

Summary

As is evident from the discussion above, automatic classification has been demonstrated using a variety of methods; some manual, some automated. Table 1 below summarizes the key points of the studies reviewed above for easy reference. The methods in the section, Automated Mathematical Comparison Systems, are the ones of greatest interest in this thesis. This is due to the fact that the systems described there not

only automate the classification process, but they automate the process of developing the representations of the underlying classification scheme as well. The classification method described by Cheng and Wu appears very promising in that it uses a simple yet effective calculation to determine document classification.

Table 1. Summary Matrix of Document Classification Studies

AUTHOR(s)	TITLE	TEXT ANALYSIS		CLASSIFICATION DETERMINATION	
		Manual	Automated	Knowledge Based	Mathematical Comparison
Savic	Designing an Expert System for Classifying Office Documents	X		X	
Cosgrove & Weimann	Expert System Technology Applied to Item Classification	X		X	
Bhatia, Deogun & Raghavan	Formation of Categories in Document Classification Systems		X	X	
Losee & Haas	Sublanguage Terms: Dictionaries, Usage, and Automatic Classification		X		X
Larson	Experiments in Automatic Library of Congress Classification		X		X
Cheng & Wu	ACS: an Automatic Classification System		X		X

III. Method

Introduction

The second stated objective of this thesis effort was to develop and propose a technique for automatically analyzing records in order to assign appropriate classification and disposition within the USAF. This chapter begins by introducing the Records Analysis and Classification System (RACS), a *proof of concept* system which was developed to meet this objective. Included with this is a brief discussion of the diagramming technique used in portraying the system. Following that are sections which describe in detail the processes which occur within RACS while classifying a new record. A discussion on the procedure used to test the operation of RACS including specific details on the sample of records collected is presented. The chapter concludes by summarizing the statistical techniques employed to analyze the performance of RACS.

RACS Introduction

The RACS program was developed using the C programming language. RACS contains many administrative functions designed to manage the various data files generated and used during program execution; an in-depth discussion of these functions is outside of the scope of this chapter. Those interested in these details can refer to Appendix B, Overview of the RACS System, and Appendix C, which contains a complete listing of the source code for the RACS program. The discussions in subsequent sections will focus on the NLP functions which constitute the heart of the RACS approach to automatic classification of USAF records.

Data Flow Diagramming. It is appropriate at this point to briefly describe the diagramming technique used in portraying RACS' operation. Data Flow Diagrams (DFD) are a graphical diagramming technique used to depict the flow and transformation of data through a set of processes (Kendall & Kendall, 1995:229). The four basic symbols used in creating DFDs are illustrated in Figure 2.

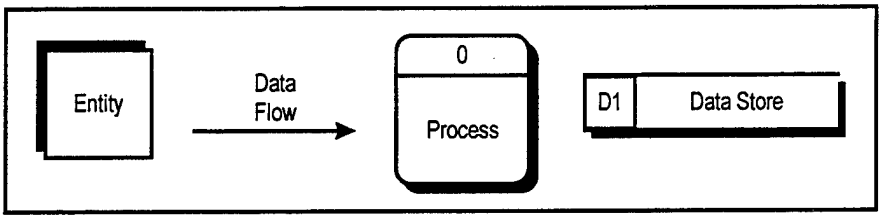

Figure 2. Four Basic Symbols for Data Flow Diagrams

Kendall and Kendall describe the four basic symbols as follows (1995:232-233):

1. An *entity* is something external to the system which can send data to and receive data from the system.
2. A *data flow* depicts the movement of data within the system.
3. A *process* transforms data as it flows through the system.
4. A *data store*, represents a repository in which data can be stored and retrieved.

The highest level DFD used in describing a system is the *Context Diagram*. This DFD contains only one process which represents the entire system and illustrates the relationships between the system and any external entities.

The second level DFD is referred to as *Diagram 0*. Diagram 0 is an exploded view of the system depicted in the Context Diagram and can show up to nine numbered

processes. In turn, each of the processes depicted in Diagram 0 can be exploded into *child diagrams* as necessary to present greater detail.

Classify New Record Context Diagram

The actual portion of the RACS system of interest in this thesis is that portion which determines the correct classification for a new record. Figure 3 presents the Context Diagram for the *Classify New Record* process. This process receives from the system user the pertinent data about the record to be classified. This data about the record, or *Record Metadata*, is used by the Classify New Record process to return the *Correct Record Class* (record classification) to the user. The concepts, *record metadata* and *record class*, will be described in the following sections.

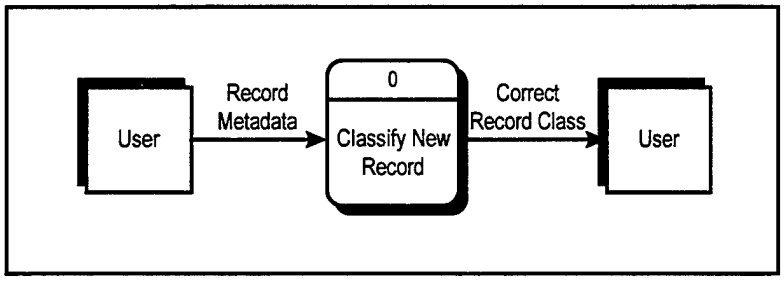

Figure 3. Classify New Record Context Diagram

Classify New Record Diagram 0

Figure 4 is the Diagram 0 DFD for the Classify New Record process.

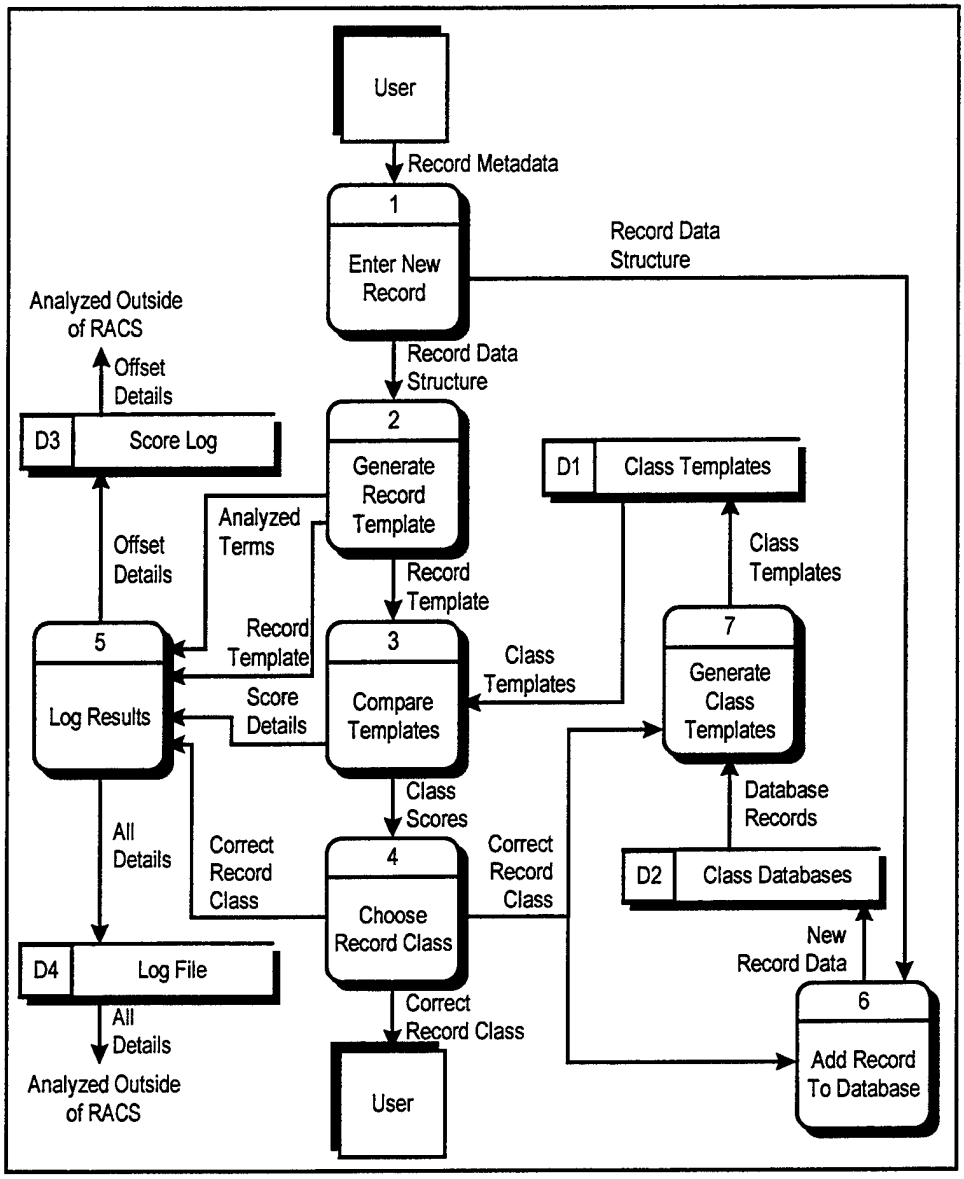

Figure 4. Classify New Record Diagram 0

As one can see from Figure 4 there are seven key processes which are involved in the Classify New Record process. As was depicted in Figure 3 (the Classify New Record Context Diagram) the Classify New Record process depicted in Figure 4 begins when a user enters the Record Metadata on a new record (shown at the top of the figure). The

process concludes when a Correct Record Class determination has been returned to the user (shown at the bottom of the figure). The seven key processes depicted in Figure 4 are described in greater detail in the following sections.

Process 1: Enter New Record

DoD-STD-5015.2 specifies nine types of metadata which any RMA must be capable of recording about a given record prior to its classification and filing. The nine types of metadata and their descriptions are listed in Table 2.

Table 2. Record Metadata Specified by DoD-STD-5015.2

Field Name	Description
Subject	The principal topic addressed in a record.
Data of Record	The date and time the record is filed by the RMA.
Addressee(s)	The name of the organization or individual to whom a record is addressed.
Media Type	The material/environment on which information is inscribed (e.g., microform, electronic, paper).
Record Format	The logical structure of a record (e.g., WordPerfect 5.2®, Microsoft Excel 4.0®). Applicable primarily to electronic records.
Location of Record	The physical location of the record. For example an operating system path-file name for an electronic record or the location of a file cabinet for a paper record.
Document Creation Date	The date and time that the author-originator created the record.
Author or Originator	The author of a document is the physical person or the office/position responsible for the creation of the record.
Originating Organization	Official name or code that reflects the office responsible for the creation of a record.

(Adapted from DoD, 1996; Prescott and others, 1995)

For this thesis project, it was assumed that the processes performed by the RACS system would in fact be just one piece of a larger RMA. With this assumption in mind, the user enters the pertinent data into the system from the Classify New Record Data

Entry interface (see Figure 31 in Appendix B). The specific data fields capable of being collected by RACS mirror those described in Table 2 with the following exceptions:

1. RACS does not collect Location of Record data. The reason for this exclusion is that a new record's location is only relevant for retrieval purposes after the record has been classified and filed. This process is outside of the scope of the RACS program so was therefore not included.

2. The user does not enter the Date of Record, RACS enters this information automatically.

3. A field called Record Type was added which is intended to capture information about the type of record being classified; for example, if a record is being filed which is an AF Form 55, AF Form 55 would be entered as the record's Record Type. This field was added for purposes of testing the system because it was felt that this type of metadata (though not required by the DoD standard) might be an important distinguishing characteristic of a given record.

Table 3 shows the eight data fields used by RACS as record metadata. The order of the individual fields for each type of metadata have been reordered to correspond to the sequence in which the user enters them (see Figure 31 in Appendix B).

Table 3. RACS Record Metadata Fields

Field Name	Enter Data	Description
Addressee(s)	Optional	The name of the organization or individual to whom a record is addressed.
Originating Organization	Mandatory	Official name or code that reflects the office responsible for the creation of a record.
Subject	Mandatory	The principal topic addressed in a record.
Author or Originator	Optional	The author of a document is the physical person or the office/position responsible for the creation of the record.
Document Creation Date	Optional	The date and time that the author-originator created the record.
Record Type	Mandatory	The type of record being entered (e.g., official memorandum, AF Form 55).
Media Type	Mandatory	The material/environment on which information is inscribed (e.g., microform, electronic, paper).
Record Format	Mandatory	The logical structure of a record (e.g., WordPerfect 5.2®, Microsoft Excel 4.0®). Applicable primarily to electronic records.

The fields labeled as *Mandatory* are used by RACS to determine the record class of a new record. These fields were selected because, as a group, they are likely to be capable of distinguishing one record from another. In contrast, the other fields, while useful for retrieval in an RMA, would probably not be descriptive enough to distinguish one record from another. For instance, it was assumed that the metadata in the Subject field would be more useful for classifying a record than the metadata in the Addressee field.

The output of the Enter New Record process is a *Record Data Structure* containing the data entered by the user into each of the fields listed in Table 3.

Process 2: Generate Record Template

As illustrated in Figure 5 below, the Generate Record Template process takes as its input the Record Data Structure created in Process 1 and performs an analysis of the mandatory record metadata to derive a representation of the original document or record called a *Record Template*.

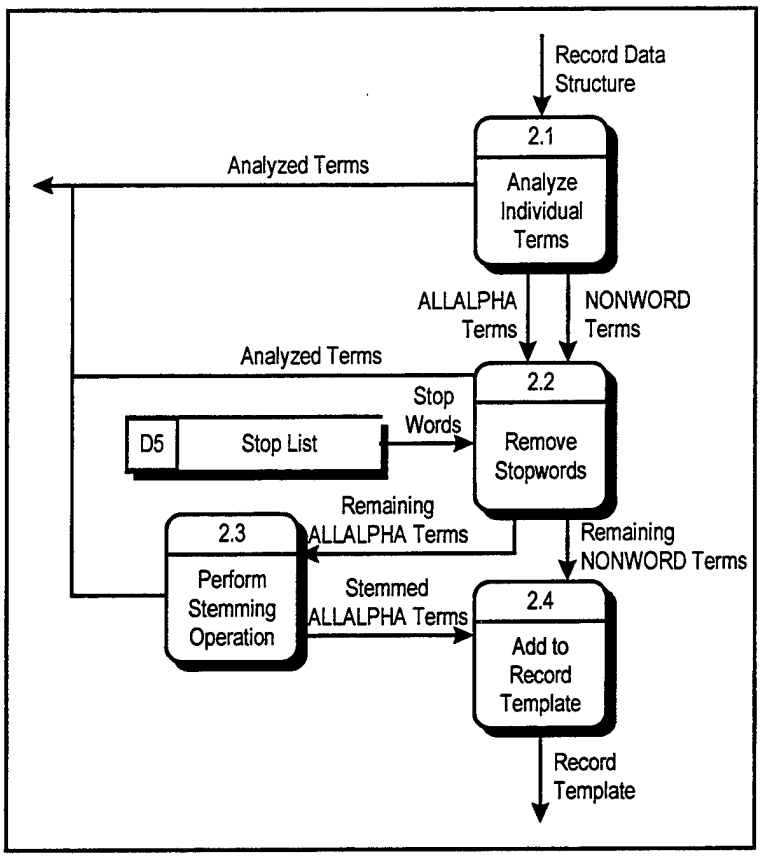

Figure 5. Generate Record Template Child Diagram

Process 2.1: Analyze Individual Terms. This sub-process performs the crucial task or extracting and classifying individual lexical terms (i.e., words) from each of the five mandatory fields of metadata in the Record Data Structure. The algorithm developed to accomplish this task is an adaptation of a lexical scanner described by Atkinson and Atkinson (1990:382-393). The algorithm uses white space and punctuation characters as delimiters when extracting individual terms for analysis. Each extracted term is classified as either ALLALPHA or NONWORD. Any punctuation encountered is classified as PUNCT. Table 4 outlines the logic used to make these distinctions. Note that the rules in this table are executed in sequence and the first rule to be found true causes the

algorithm to assign that classification to the current term and then begin executing the rules again with a new term.

Table 4. Analyze Individual Terms Logic Rules

RULE	IF first character is	AND second character is	AND all other characters are	THEN term type is
1	A-Z or a-z	a-z	a-z	ALLALPHA
2	A-Z	A-Z	A-Z or a-z	NONWORD
3	0-9	0-9 or / or -	0-9 or / or -	NONWORD
4	A-Z or a-z	N/A	N/A	ALLALPHA
5	0-9	N/A	N/A	NONWORD
6	any punctuation	N/A	N/A	PUNCT

Key: A-Z uppercase alphabet character
a-z lowercase alphabet character
0-9 numerical character

To illustrate the use of the rules in Table 4, consider the following terms extracted from a Subject metadata field: Administrative, AFR, and 177-16. The term "Administrative" would be classified as ALLALPHA because it meets the conditions in Rule 1. In contrast, the term "AFR" would be classified as NONWORD because it does not meet the conditions in Rule 1 but does meet those in Rule 2. Likewise, the term "177-16" would be classified as NONWORD because, although it does not meet the conditions in either Rule 1 or Rule 2, it does meet the conditions in Rule 3.

All PUNCT is eliminated from further analysis while the terms classified as ALLALPHA and NONWORD are converted to all lowercase characters and then sent for further analysis to Process 2.2, Remove Stopwords.

Process 2.2: Remove Stopwords. During this process each term from Process 2.1 is compared to a stoplist. If one of the terms from the record matches a stopword in the

stoplist the term is marked as a stopword and dropped from further analysis. The stoplist used with RACS was presented in Fox and contains 425 common English words (e.g., "a," "the," "and," "of," etc.) (1992:114-115). Thus, those terms which are assumed to have relatively little value in analyzing the meaningful differences among records are removed from consideration. The only additional stopwords added to the stoplist were the terms "af" and "form". These additional stopwords were added due to the fact that they occurred frequently in all types of records from various record classes in the sample collected for this thesis and consequently were not considered valuable in discerning meaningful differences among records. The complete stoplist used in this thesis can be found in Appendix E.

Following removal of all stopwords, the remaining ALLALPHA terms are sent to Process 2.3, Perform Stemming Operation, while all remaining NONWORD terms are sent directly to Process 2.4, Add to Record Template.

Process 2.3: Perform Stemming Operation. The stemming algorithm used in RACS was presented in Fox (1992: 151-160) and is an adaptation of a suffix stripping algorithm proposed by Porter (1980). Porter's algorithm works by "treating complex suffixes as compounds made up of simple suffixes, and removing the simple suffixes in a number of steps" (1980:130). The result of this stemming operation is a list of terms which have been reduced to a common morphological stem. These common stems enhance the ability of RACS to match related terms which would have otherwise appeared to be different. For example, the stemming process would take as its input the terms "connect," "connected," "connecting," "connection" and "connections" and reduce each

term to the common stem "connect." Only ALLALPHA terms are subject to stemming in this system. A NONWORD term (e.g., "AFR") is assumed not to share a common stem with any other NONWORDs.

Process 2.4: Add to Record Template. The final process within the general process Generate Record Template takes as its input all remaining NONWORD terms after stopword removal and all remaining ALLALPHA terms after stopword removal and stemming. These terms are placed with their frequency of occurrence in the metadata into a record template which is RACS' representation of the document being classified. Each term is added to the appropriate term array in the record template; in other words, terms which were extracted from the Subject field in the original Record Data Structure are added to a Subject Array in the Record Template and terms from the Media Type field are added to the Record Template's Media Type Array. Refer to Appendix F to see an example of a record template as well as the results of the analysis processes in Process 2 on one record.

Process 3: Compare Templates

Recall from Figure 4 that the Compare Templates process takes as its input the Record Template discussed in the previous section and each of the five Class Templates in turn. *Class templates* are identical to record templates except for the fact that instead of representing one record, a given class template represents all records previously added to that class (i.e., a record category).

Thus, for each record template to class template comparison there are five distinct arrays of terms to be compared; Subject, Originating Organization, Record Type, Media

Type, and Record Format. To determine the amount of overlap (similarity) between a given record template array and the corresponding class template array, a Modified Overlap Coefficient (MOC) is calculated. To illustrate, the formulas for calculating a MOC indicating the overlap between the record template's subject array and class template 1's subject array are defined below (adapted from Cheng and Wu, 1995:293).

Let C_{1Sub} in Equation 1 represent class template 1's subject array and R_{Sub} in Equation 2 represent the record template's subject array:

$$C_{1Sub} = \{(c_1, f_1), (c_2, f_1), \ldots, (c_i, f_i), \ldots, (c_m, f_m)\} \quad (1)$$

$$R_{Sub} = \{(r_1, g_1), (r_2, g_1), \ldots, (r_j, g_j), \ldots, (r_n, g_n)\} \quad (2)$$

where

c_i = term i in C_{1Sub}

f_i = frequency of term i

m = the number of terms in C_{1Sub}

r_j = term j in R_{Sub}

g_j = frequency of term j

n = the number of terms in R_{Sub}

Then the formula for calculating the MOC of the record template subject array with class template 1's subject array is:

$$MOC_{1Sub} = \frac{\sum e_{ij}}{N_1(G)} \tag{3}$$

where

$e_{ij} = f_i(g_j)$ if $c_i = r_j$

N_1 = number of records in class template 1

$G = \sum g_i$ if $c_i = r_j$

In all, 25 MOC values for a given record template are calculated; there are five different array MOCs (representing overlap of terms in the subject field, the originating organization field, etc.) for each of the five class templates (representing record class 1, record class 2, etc.). Five composite class scores are calculated by summing the five MOC values from each record template to class template comparison.

Process 4: Choose Record Class

This process represents the point at which the user is presented with a rank ordered list of the most likely classifications for the record currently being classified. Figure 33 in Appendix B is a depiction of the interface the user actually sees. The user is required to enter the correct record class for the current record. Once this is done, the output of the process, Correct Record Class, flows to the last three processes included within the larger process, Classify New Record (see Figure 4). As far as the user is concerned, RACS is now ready to classify another record.

Process 5: Log Results

This important process takes as its input the results of various other processes within the overall Classify New Record process and records them to one of two log files which are used for data analysis. Another key function performed within the Log Results process is the calculation of an *offset* which is in essence an error value. The offset indicates how far off RACS was from determining the correct record class for the current record.

An examination of Figure 6 reveals the interrelationships between the various inputs and the three sub-processes involved.

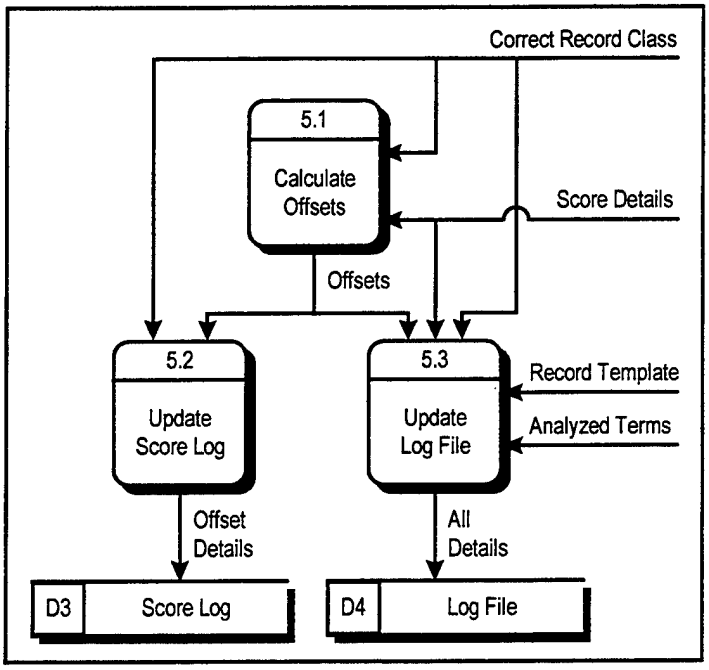

Figure 6. Log Results Child Diagram

Process 5.1: Calculate Offsets. The method for calculating offsets is tied to the ranks RACS assigns to the five classes during the Compare Templates process. For example, if the correct class (as entered by the user) for a new record was class 4 and RACS had determined that class 4 was the third most likely classification, then RACS was off by two positions in ranking the correct class and the offset is 2. If for the same record, RACS had determined that class 4 was the first most likely classification (rank 1) then the offset would be 0.

There is an exception to the general rule used in calculating offsets when RACS assigns the same score and rank to two or more record classes. For example, if class 5 is the correct classification and RACS ranks class 5 and class 1 as tied for most likely classification, then RACS could not distinguish between class 1 and class 5 and the offset is calculated as 1 (rather than 0) due to the ambiguity.

Process 5.2: Update Score Log. This process takes the Offsets calculated during the Calculate Offsets process and the Correct Record Class from the Choose Record Class process and adds the values to the simple log file Score Log which is depicted by data store D3 in Figure 6. Appendix G is an excerpt from an actual Score Log.

Process 5.3: Update Log File. As Figure 6 illustrates, this process takes as its inputs the results of various processes and adds the data to the Log File (data store D4). Appendix F is an excerpt from an actual Log File and illustrates the various information captured.

Process 6: Add Record To Database

As is indicated in Figure 4, this simple process adds the data from the original Record Data Structure for the current record to the database corresponding to the Correct Record Class; in other words, if the current record belongs to class 3, the Add Record To Database process would add this record's data to the database containing data on class 3.

Process 7: Generate Class Templates

Once a record has been classified and its metadata has been added to the appropriate class database, the corresponding class template is regenerated. As mentioned earlier, a class template is identical to a record template except for the fact that a class template is a representation of *all* the records previously added to that class. An examination of the processes illustrated in Figure 5 and the processes illustrated in Figure 7 emphasize the fact that record templates and class templates are virtually identical. Once again, the only difference is the fact that each record stored in a given Class Database (signified by data store D2 in Figure 7) is analyzed and used to build the corresponding Class Template (signified by data store D1 in Figure 7). Appendix H contains a sample of a typical class template.

The strength of this approach is that RACS is in essence capable of learning in that, as more records are added to a given class, its knowledge of the records typically found in that class increases. The converse to this is the fact that a record entered into a class to which it does not belong can distort RACS' representation of a given class.

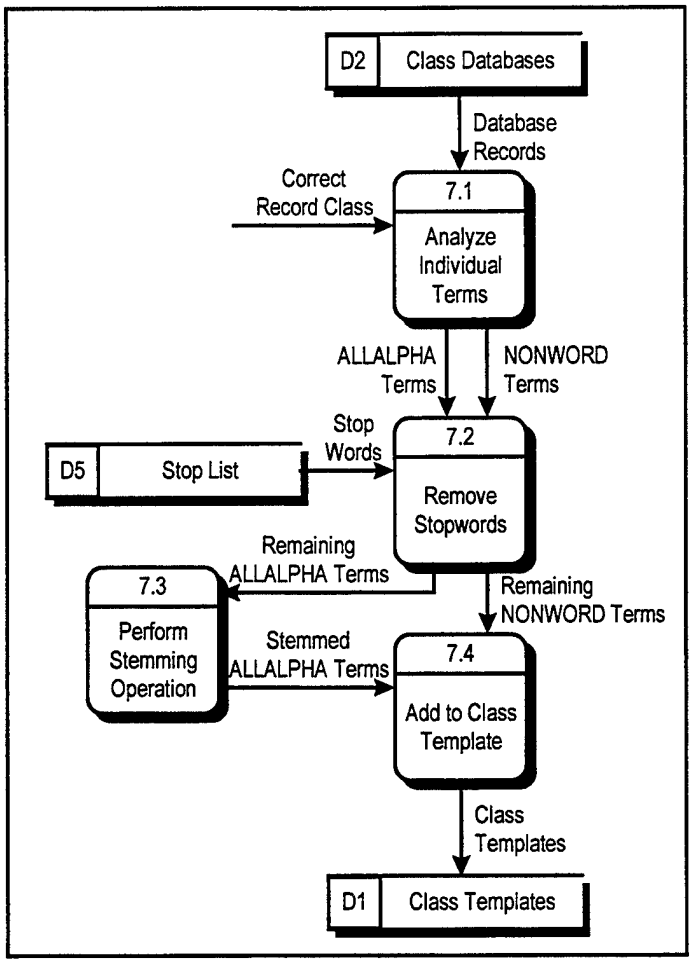

Figure 7. Generate Class Templates Child Diagram

Procedure for Testing RACS

The third and final objective of this thesis was to demonstrate the classification techniques developed on a limited set of sample records. Three specific questions were formulated which served as the basic requirements for designing the actual testing procedures. The questions were as follows:

1. How accurately does RACS classify records and is it capable of learning?
2. Since RACS was designed to be a "learning" system, does the order in which records are added to RACS' record classes affect overall classification accuracy?

3. Does the weighting of the five fields of metadata used for scoring affect overall classification accuracy?

The discussion which follows presents an overview of the sample records collected for this thesis. Following that is a discussion of the actual procedure employed to test RACS.

Sample Records. The 88th Support Group Administration Office (88 SPTG/CCE) was chosen as the source of the sample records for this thesis. There were two primary reasons for the selection of the 88 SPTG/CCE. The first reason is that the files plan for the 88 SPTG/CCE consisted of 23 rules, 19 of which are found among the "Common Tables and Rules" in Appendix I. As stated in Chapter I, there are over 6000 disposition rules in AFMAN 37-139. Of these, a relatively small number of rules are common for virtually all files plans across the USAF (Bolden and Pollard, 1996). Appendix I is an adaptation of a table provided by the personnel in the Base Records Management office at Wright-Patterson AFB, OH. The table lists the common tables and rules for files plans on Wright-Patterson.

Second, a Support Group Administration Office is an organization which can be found on nearly all USAF Bases. The proceeding two factors taken together demonstrate that the files plan in use by the personnel at the 88 SPTG/CCE might be considered representative of a typical USAF files plan.

The files plan for the 88 SPTG/CCE, illustrated in Appendix J, contained record classes corresponding to 23 distinct disposition rules. AFMAN 37-123 allows for subdivisions to be added to files plans for ease of filing (SECAF, 1994b:3.2). It should

be noted that while the major items/disposition rules contained in a given files plan are governed by AFMAN 37-139, subdivisions are not; a given organization can include whatever subdivisions they deem appropriate to meet their specific needs. Subdivisions are illustrated in Appendix J in items 4, 6, 7, 17, 20, and 23. For filing purposes, each of these rules and subdivisions correspond to a physical file folder in the 88 SPTG/CCE's official files.

To demonstrate the operation of RACS, five record classes or categories were selected for sampling (see Table 5). The following factors were considered when the record classes were selected.

1. At least two record classes should be subdivisions (i.e., would be determined by the individual office rather than USAF regulation) in order to demonstrate RACS' ability to be customized to the needs of any given office.

2. At least one record class should contain records of a homogeneous nature. In other words, all of the files in the class are a specific document type (such as a single USAF form).

3. Several record classes should contain records of a heterogeneous nature in order to test RACS' ability to classify diverse records (such as forms, memorandums, etc.) into the same class.

4. The number of records physically filed in the file folder corresponding to a given record class should be at least five in order to provide a sufficient sample for testing RACS.

Table 5. Record Classes Selected for Sampling

RACS Class	# Rcrds	Item	Title	Disposition Rule
1	26	3	Delegations/Designations of Authority & Additional Duty Assignments	T 11-02 R21.00
2	30	6-3-2	Office Administrative Files - Internal Admin and Housekeeping - Supplies/Equipment	T 11-01 R01.00
3	5	6-4	Office Administrative Files - Internal Admin and Housekeeping - Safety	T 11-01 R01.00
4	13	12	Internal Inspections/Self-Inspection Check Lists/Inventories	T 11-02 R33.00
5	39	15	Suggestions, Inventions, & Scientific Achievements - At Evaluating Office	T900-02 R02.00

In all, data on 113 records were gathered. The actual data (i.e., record metadata) shown in Table 3 was compiled during a review of each sample record located in the physical file folders of the 88th Support Group. A complete listing of the sample records used in this thesis can be found in Appendix K.

Determining the Effects of Record Order. In order to test the effects of record entry order on RACS' classification accuracy, the following procedure was employed. Each sample record was assigned a random number using a computer-based random number generator which was seeded by the time from a personal computer's internal clock. The list of records was then ordered according to the random numbers. This procedure was then repeated on the same personal computer resulting in two randomly ordered lists of records.

Determining the Effects of Various Weighting Schemes. Three different weighting schemes were employed to score every record entered for classification (see Table 6). The column labels in the top row of the table signify the MOC value for the record

metadata field listed as the subscript. The row labels listed in the first column are the notations for each weighting scheme.

Table 6. Class Score Weighting Schemes

	MOC_{Sub}	MOC_{Org}	MOC_{Typ}	MOC_{Med}	MOC_{Frm}
20/20/20/20/20	0.2	0.2	0.2	0.2	0.2
30/20/30/10/10	0.3	0.2	0.3	0.1	0.1
50/30/00/10/10	0.5	0.3	0.0	0.1	0.1

Key: Sub = Subject Field
Org = Originating Organization Field
Typ = Record Type Field
Med = Media Type Field
Frm = Record Format Field

The rationale for the selection of the three weighting schemes was as follows:

1. 20/20/20/20/20 - This scheme assigned equal weight to all applicable data fields. This scheme was implemented to provide a standard against which the other two weighting schemes could be compared in terms of classification accuracy.

2. 30/20/30/10/10 - It was felt that the Subject and Record Type fields would be of more value in distinguishing correct record class than the other fields. Therefore, under this scheme the Subject and Record Type fields were given greater weight than the other three fields.

3. 50/30/00/10/10 - Record Type does not factor into the score calculated using this scheme. This scheme was designed to provide insight as to the effect of the addition of the field, Record Type, which is not mandated by the DoD standard.

Conducting the Test. The procedure for conducting the actual test was straightforward. All of RACS' data files were cleared of data and the first set of randomly ordered records was entered into the system. After the log files were saved to an alternate location, the data files were once again cleared and the procedure was repeated with the second set of randomly ordered records.

Analyzing RACS' Performance.

The three offset values (corresponding to the three weighting schemes) recorded for each sample record classified served as the raw data which was analyzed to determine RACS' accuracy as an automated records analysis and classification system. The following sections summarize the analysis conducted to answer the three research questions.

Question 1. **How accurately does RACS classify records and is it capable of learning?** To analyze RACS' overall accuracy at classifying records, relative frequency histograms were developed (McClave and Benson, 1994:28-32). Time series plots including exponentially smoothed trend lines (McClave and Benson, 1994:796-798) were prepared to illustrate the "learning curve" associated with each set of randomly ordered sample records and each weighting scheme.

Question 2. **Since RACS was designed to be a "learning" system, does the order in which records are added to RACS' record classes affect overall classification accuracy?** Paired sample *t* tests (McClave and Benson, 1994:420-424) were conducted to determine if there was a statistically significant difference between the offsets generated by each set of randomly ordered sample records. This test was chosen for two primary reasons. First, the samples in this case were related (i.e., the exact same set of records are used twice). Thus, since the samples were not independent a standard two-sample *t* test was not appropriate. Second, the standard assumptions for a paired difference test of hypothesis were met with the offset data.

Question 3. **Does the weighting of the five fields of metadata used for scoring affect overall classification accuracy?** Within each set of randomly ordered sample records, Wilcoxon signed rank tests for a paired difference experiment (McClave and Benson, 1994:935-940) were conducted to determine if there was a statistically significant difference between the offsets generated by each weighting scheme.

Summary

In order to meet the objectives set forth in the thesis effort, automated classification techniques were developed and implemented in a proof of concept system, RACS. This system, along with the procedures for testing and analyzing the operation of RACS were explained in detail. The following chapter presents a detailed analysis of the results from the tests conducted.

IV. Results and Analysis

Introduction

As has been discussed previously, the three offset values (corresponding to the three weighting schemes) were recorded for each sample record classified. This chapter provides detailed analysis of these offset values (the actual raw data can be found in Appendix L). For clarity of discussion, this chapter is subdivided into a series of sections corresponding to the three research questions posed in the previous chapter.

Question 1

How accurately does RACS classify records and is it capable of learning? There are essentially two pieces to this question which were analyzed using separate techniques. The first portion of the question is concerned with an overall picture of RACS' accuracy while the second portion is concerned specifically with determining if RACS is in fact capable of learning.

To analyze RACS' overall accuracy while classifying records, simple relative frequency histograms were utilized to illustrate the results of the classification tests. Specifically, two histograms were developed; the histogram illustrated in Figure 8 presents the results of the tests conducted using the first randomly ordered set of records and Figure 9 presents the results of the tests using the second randomly ordered set of records. The values on the horizontal axis in each histogram correspond to each possible offset value while the vertical axis represents the number of records which resulted in a

given offset value. The three different series of vertical bars correspond to the three weighting schemes used.

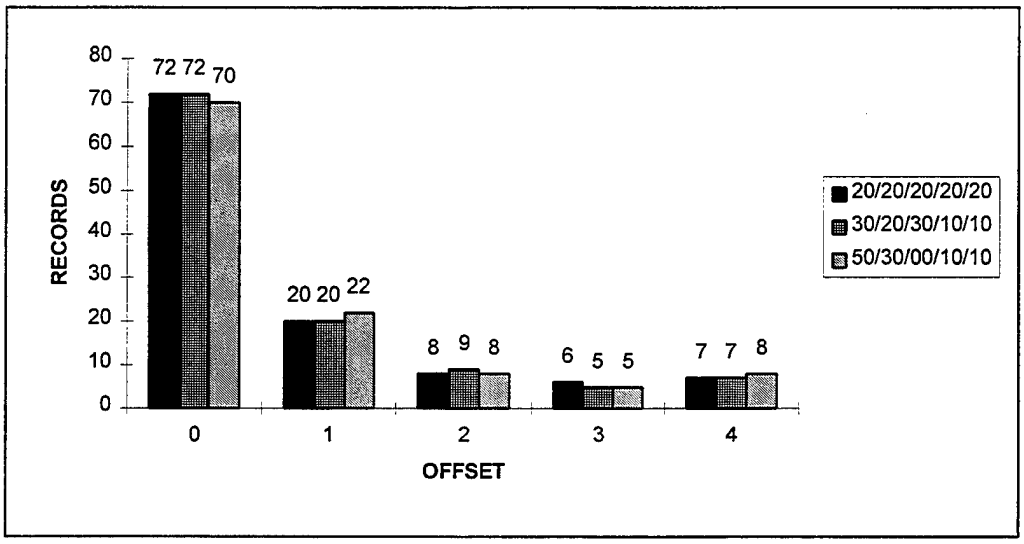

Figure 8. Histogram of Sample 1 Results Showing the Distribution of Offset Values for Each of the Three Weighting Schemes for Calculating Class Scores

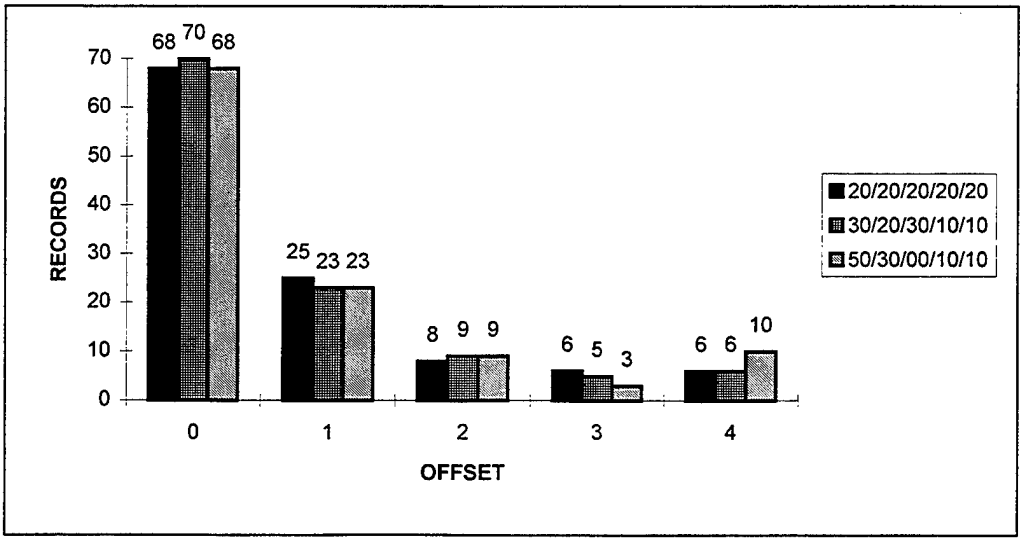

Figure 9. Histogram of Sample 2 Results Showing the Distribution of Offset Values for Each of the Three Weighting Schemes for Calculating Class Scores

A visual examination of the two histograms presented indicates that RACS was able to correctly classify (i.e., achieve an offset of 0) 70 out of 113 records on average. Additionally, RACS was able to classify an additional 22 records with an offset of 1. While this data is extremely useful for illustrating the overall performance of RACS, the analysis conducted to determine RACS ability to learn provides more detailed and rigorous insight into RACS' ability to accurately classify records.

As discussed earlier, RACS was designed to be a learning system. Before any records have been classified RACS knows nothing about the particular record classes of interest. As records are added to a given class, RACS knowledge of the types of records typically contained in that class increases.

If RACS were only capable of guessing a given record's classification, one would expect that there would not be any evidence that learning had occurred and that the offsets produced would occur in a purely random fashion. Figure 10 illustrates this hypothetical situation.

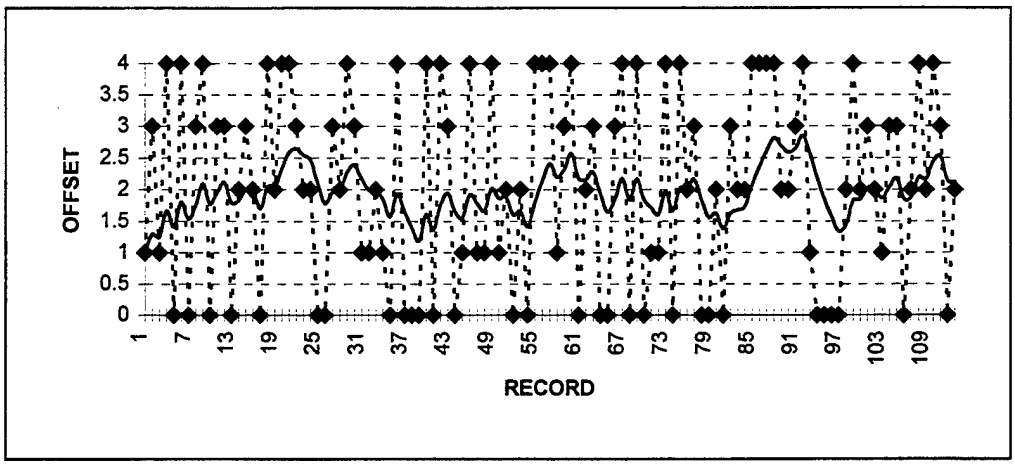

Figure 10. Example of the Accuracy of Record Classification in a Random or "Guessing" System

The dashed lined in the figure is a time series plot of the offset values computed for each sequential record classified (the offset values in Figure 10 were generated randomly for purposes of illustration only). The solid line is an exponentially smoothed line which is a smoothed version of the offset line and is intended to indicate the general trend in the data.

In contrast to the results of the hypothetical system depicted in Figure 10, the actual plots generated from the tests conducted with RACS indicate that learning did in fact occur. The following six figures illustrate the results of the tests conducted for each of the three weighting schemes in both randomly ordered sets of records.

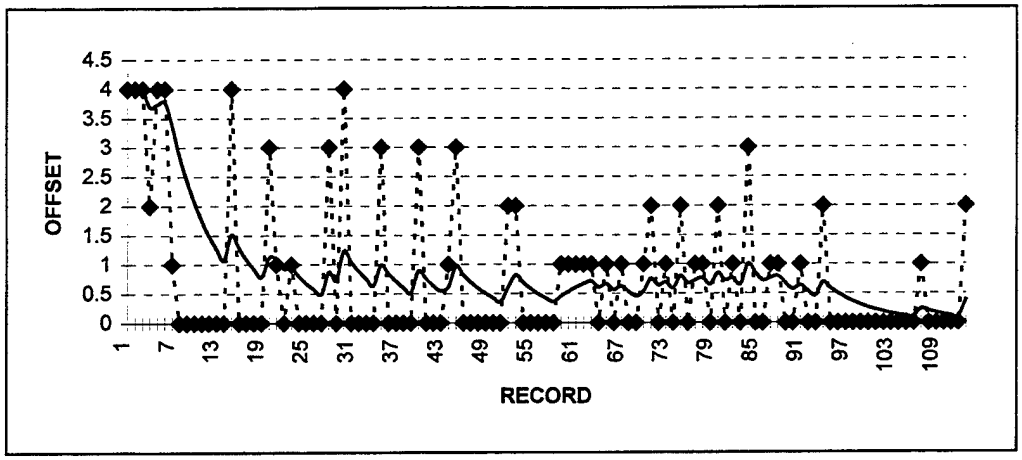

Figure 11. Sample 1 - Time Series Results for RACS with Weighting Scheme 20/20/20/20/20

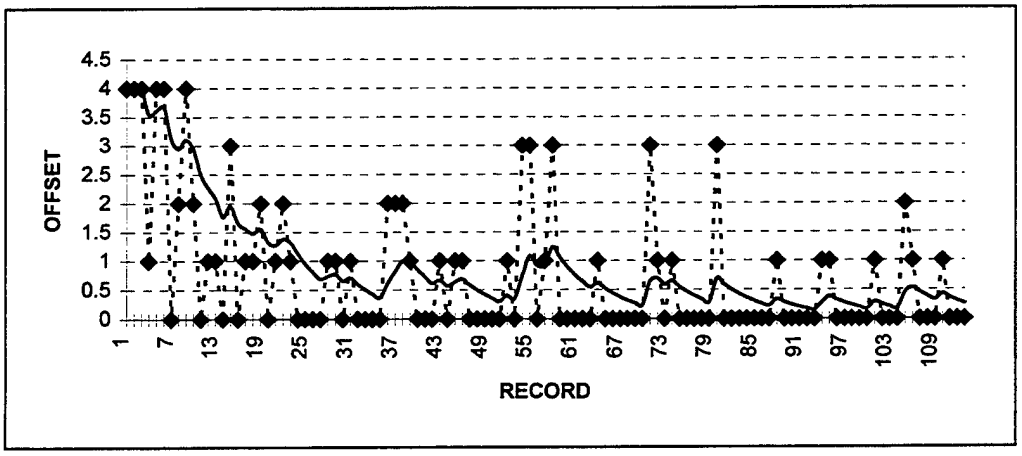

Figure 12. Sample 2 - Time Series Results for RACS with Weighting Scheme 20/20/20/20/20

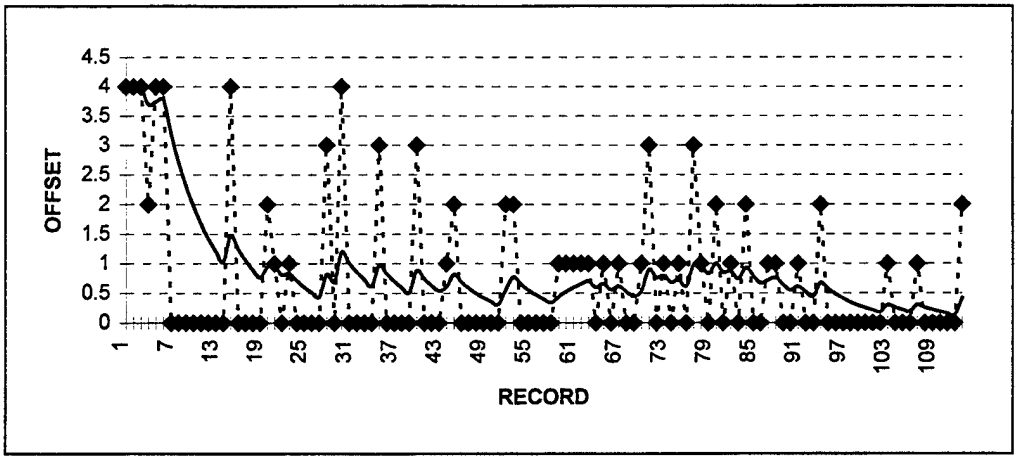

Figure 13. Sample 1 - Time Series Results for RACS with Weighting Scheme 30/20/30/10/10

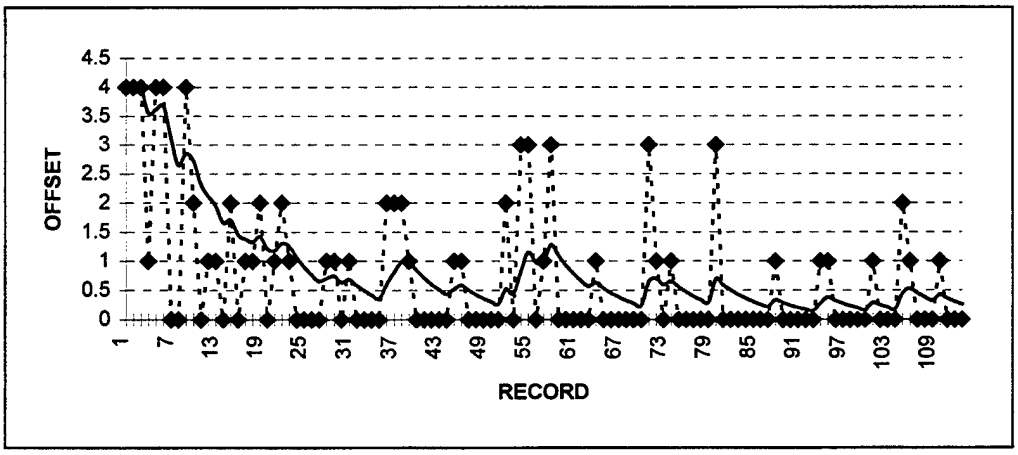

Figure 14. Sample 2 - Time Series Results for RACS with Weighting Scheme 30/20/30/10/10

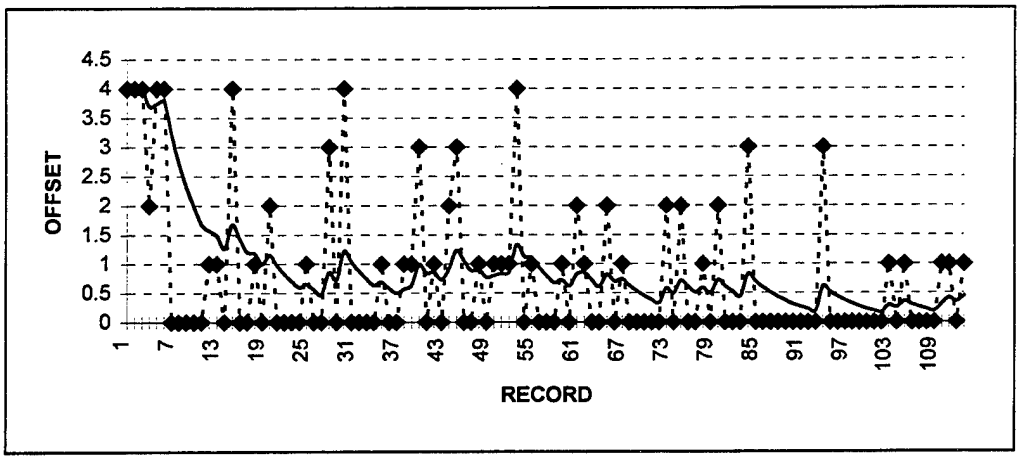

Figure 15. Sample 1 - Time Series Results for RACS with Weighting Scheme 50/30/00/10/10

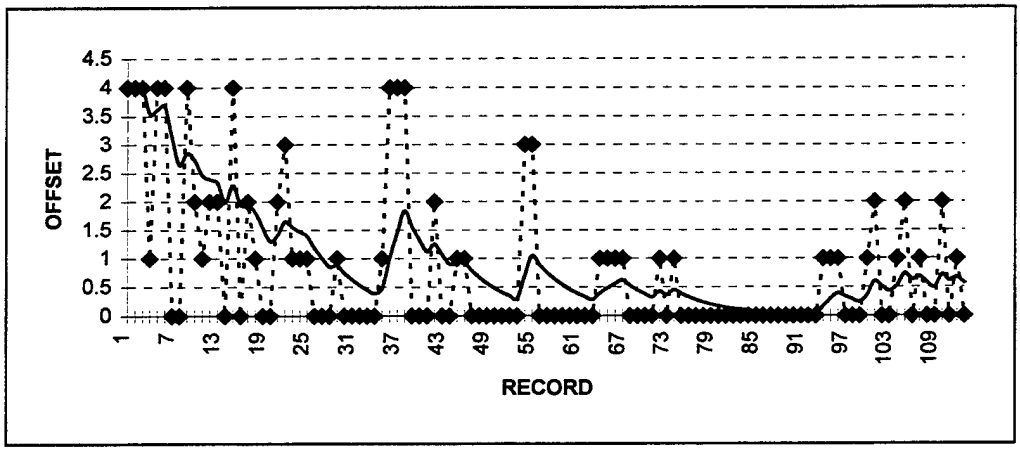

Figure 16. Sample 2 - Time Series Results for RACS with Weighting Scheme 50/30/00/10/10

Figure 11 through Figure 16 provide visual evidence that RACS' classification accuracy improved over time; in other words, RACS did indeed learn during the process of classifying the 113 records in each sample. This is indicated by the fact that there is a distinguishable downward trend in the exponentially smoothed line (the solid line in each figure).

Question 2

Since RACS was designed to be a "learning" system, does the order in which records are added to RACS' record classes affect overall classification accuracy?

Paired sample *t* tests were conducted to determine if there was a statistically significant difference between the offsets generated by the two randomly ordered sets of records (i.e., Samples 1 and 2).

Three two-tailed paired sample *t* tests were conducted, one corresponding to each weighting scheme. The hypotheses tested by each *t* test were as follows:

- Null Hypothesis: The population of offsets corresponding to a given weighting scheme from the first set of sample records does not differ from the population of offsets associated with the same weighting scheme from the second set of sample records.
- Alternative Hypothesis: The populations are in fact different; essentially indicating that order does have an effect.

Table 7 summarizes the key data associated with each of the tests conducted.

Table 7. Paired Sample *t* Test Results

	20/20/20/20/20	30/20/30/10/10	50/30/00/10/10
α	0.05	0.05	0.05
n_D	113	113	113
df	112	112	112
$\bar{x}_D$	-0.0088	0.0088	-0.0442
s_D	1.2921	1.2500	1.3976
t	0.0753	-0.0728	-0.3365
Rejection Region	$t < -1.98$ or $t > 1.98$	$t < -1.98$ or $t > 1.98$	$t < -1.98$ or $t > 1.98$
Result	Fail to reject Null	Fail to reject Null	Fail to reject Null

The results presented in the table indicate that record order did not have a statistically significant effect on classification accuracy under any of the weighting schemes.

Question 3

Does the weighting of the five fields of metadata used for scoring affect overall classification accuracy? Within each set of randomly ordered sample records, Wilcoxon signed rank tests for a paired difference experiment (McClave and Benson, 1994:935-940) were conducted to determine if there was a statistically significant difference between the offsets generated by each weighting scheme. Three Wilcoxon tests were conducted for each set of sample records such that all combinations of paired comparisons were examined. For all Wilcoxon tests, the hypotheses being tested were as follows:

- Null Hypothesis: The two sampled populations have identical probability distributions.
- Alternative Hypothesis: The probability distribution for population A is shifted to the right or to the left of that for population B.

The key values associated with each of the tests are summarized in Table 8 and Table 9 below:

Table 8. Sample 1 Wilcoxon Signed Rank Tests Results

	20/20/20/20/20 30/20/30/10/10	20/20/20/20/20 50/30/00/10/10	30/20/30/10/10 50/30/00/10/10
Cases (n)	8	38	38
T_+	16	344	320.5
T_-	20	397	420.5
T	16	344	320.5
Test Statistic (T for $n < 25$; z otherwise)	16	-0.384	-0.725
α	0.05	0.05	0.05
Rejection Region	$T \leq 4$	$z < -1.96$ or $z > 1.96$	$z < -1.96$ or $z > 1.96$
Result	Fail to reject Null	Fail to reject Null	Fail to reject null

Table 9. Sample 2 Wilcoxon Signed Rank Tests Results

	20/20/20/20/20 30/20/30/10/10	20/20/20/20/20 50/30/00/10/10	30/20/30/10/10 50/30/00/10/10
Cases (n)	4	37	35
T_+	2	287	238
T_-	8	416	392
T	2	287	238
Test Statistic (T for $n < 25$; z otherwise)	2	-0.973	-1.26119
α	0.05	0.05	0.05
Rejection Region	N/A	$z < -1.96$ or $z > 1.96$	$z < -1.96$ or $z > 1.96$
Result	N/A	Fail to reject null	Fail to reject null

All of the Wilcoxon tests conducted failed to reject the null hypothesis. This indicates that the weighting schemes utilized had no significant statistical impact on RACS' overall ability to classify records. Note that a test was not actually conducted for the first pairing in sample two. The reason for this is that, of the 113 offset pairs, only 4 resulted in a difference greater than 0 and the Wilcoxon test does not apply to samples with less than 5 cases.

One significant implication of these results is that the additional field, Record Type, does not appear to contribute significantly to the overall accuracy of classification with this sample of records. This is evidenced by the fact that the 50/30/00/10/10 weighting scheme which excludes Record Type from the calculation of a composite class score did not differ statistically from the other two weighting schemes.

Differences Among Individual Record Classes

The focus of the analysis conducted for this thesis was on RACS' accuracy from a whole system perspective. While this remains the perspective of greatest interest, some observations were made during the course of this thesis study about RACS' accuracy within individual record classes. Appendices M through Q contain an exhaustive set of graphs illustrating the results of the tests for each of the five record classes used in this thesis.

Of particular interest are the graphs for record class two (see Appendix N). The graphs provide evidence that RACS was not particularly successful at classifying records from this category. A review of the records contained in that class as well as the results stored in the various log files seems to indicate that the diversity of the records (i.e., record class two included two different types of forms as well as a variety of official memorandums with diverse subjects) stored in this particular class degraded RACS' ability to accurately classify its records.

Summary

This chapter presented an in-depth analysis of the tests conducted with the RACS proof of concept system. The three specific research questions which were proposed in Chapter III served as the framework within which the results were presented. Essentially, the results indicate that RACS is an effective system for classifying records and that it is capable of learning over time. The results also indicate that the various weighting schemes employed did not have a significant impact on the overall accuracy of the system. The next chapter presents the conclusions of this author and outlines some potential areas for future research.

V. Conclusions and Recommendations

Introduction

The basic purpose of this thesis effort was to develop and demonstrate techniques for the automatic classification of USAF records using a computer based system. There were three basic objectives established which needed to be met in order to solve this problem. The first several sections of this chapter summarize the actions taken to meet these objectives. Following that, recommendations as to areas which warrant further research are presented. The last section in this chapter presents this author's final conclusions as to the feasibility of automatic analysis and classification of USAF records

Research Objective 1

Locate and summarize the various automatic document classification techniques being employed by researchers and practitioners on related projects throughout the world.

Chapter II described some key concepts relevant to the process of automated analysis and classification of documents. Additionally, the chapter provided an overview of six relevant classification projects reported in the literature. The study presented by Cheng and Wu (1995) outlined some of the key techniques such as the Modified Overlap Coefficient which were incorporated into the proof of concept system developed during this thesis research process.

Research Objective 2

Develop and propose a technique for automatically analyzing records in order to assign appropriate classification and disposition within the USAF.

Chapter III introduced the Records Analysis and Classification System or RACS for short. RACS is a proof of concept system developed using the C programming language to meet this objective. The chapter outlined in detail the various processes and techniques which were incorporated into RACS to make automatic analysis and classification possible.

Figure 17 is a repetition of the Context Diagram for the Classify New Record process.

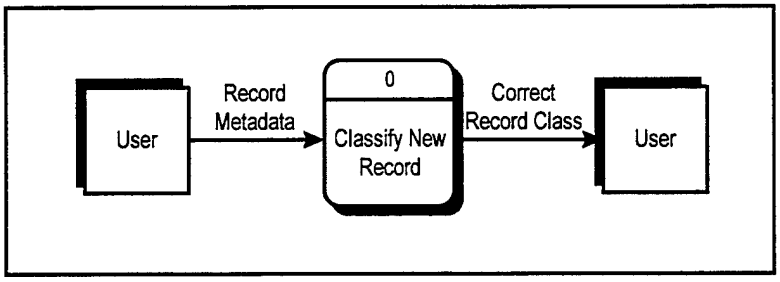

Figure 17. Classify New Record Context Diagram

The Classify New Record process begins by accepting the Record Metadata on a new record to be classified from the user. RACS then performs a series of processes with the record metadata in order to determine the Correct Record Class for the new record.

Research Objective 3

Demonstrate the proposed technique on a limited set of sample records.

A sample of 113 records from five different record classes was collected from the files of the 88 SPTG/CCE. The actual data collected about each record consisted of the record metadata which was summarized in Table 3. The sample of records was randomly ordered twice in order to produce two different sets of randomly ordered sample records

To test RACS, each randomly ordered set of sample records was entered into the system and the results were recorded. After each sample had been entered, the results of the tests were analyzed.

The analysis of the results indicated that RACS did exhibit the ability to improve its classification accuracy as more records were entered (i.e., it was capable of "learning"). It was found that the order the sample records were entered did not have a statistically significant effect on RACS' classification accuracy. The last observation was that the use of different weighting schemes did not have a statistically significant effect on RACS' classification accuracy.

Recommendations

The research conducted in conjunction with this thesis is just the first step. There are many aspects of the analysis and classification techniques incorporated in RACS which warrant further study. Some of the specific areas which are ripe for future research efforts are described below.

- **Develop a specialized USAF stoplist.**

The stoplist utilized in this thesis was a very general purpose stoplist, not at all tailored to the peculiarities of USAF records. A study of the most frequently occurring words in a large sample of USAF records' metadata could yield a stoplist more attuned to the specific needs of an automated record classification system within the USAF.

- **Investigate alternative methods for scoring record/class template comparisons; i.e., investigate alternatives to the MOC calculation.**

The MOC calculation presented in this thesis is only one of many calculation methods presented in the literature for quantifying the amount of overlap between a document and a given class of documents (see Cheng and Wu, 1995:293). The accuracy achieved by RACS in this thesis study could perhaps be improved by the utilization of a different scoring method. For example, the MOC calculation considers the frequency with which terms occurred in a new record versus the frequency with which matching terms occurred in the whole class. Perhaps a calculation technique which considered purely the number of terms in common between a new record being classified and each record class would yield the correct classification more often (i.e., result in an offset of 0).

- **Investigate alternate combinations of metadata fields and weighting schemes.**

Although the results of this study indicated that the three weighting schemes utilized did not have a significant impact on classification accuracy, this author is not convinced that weighting schemes cannot contribute to accuracy of classification. There are myriad other weighting schemes possible with the five record metadata fields used in this study. Additionally, the five metadata fields utilized in this thesis may not in fact be the best combination of fields to represent a document.

- **Investigate alternate ways to represent records in the class templates.**

RACS' representation of a particular class consisted simply of the terms extracted from the metadata fields in the records belonging to that class along with the frequency with which the individual terms occurred. Alternate methods of representing a given class could be developed and compared with the method utilized in this thesis in an attempt to find the optimal class representation method. For example, one alternative would be to represent each record in a class template individually. To determine correct classification a new record being classified would be compared to each record previously added to a given class and a similarity score would be calculated. A composite score for each record class would be determined by summing the aforementioned scores.

- **Test the operation of a system such as RACS in an actual office environment.**

This study investigated the accuracy of RACS using a limited number of record classes and a relatively small set of sample records. A valuable study to validate the results achieved in this thesis would be to implement a system similar to RACS in an actual office and analyze its performance while classifying all records handled in that office.

Conclusion

The bottom line result of this thesis effort is this; automated analysis and classification of USAF records is possible. The tests conducted with RACS demonstrated the fact that records from five distinct record classes could be classified with a reasonable level of accuracy. It is true that RACS was not perfect, but in an actual implementation,

the techniques demonstrated with RACS could serve as a powerful productivity aid to all USAF personnel who create, disseminate and store records.

Appendix A: Acronyms

AFI 37-122	Air Force Instruction 37-122 Air Force Records Management Program
AFMAN 37-123	Air Force Manual 37-123 Management of Records
AFMAN 37-139	Air Force Manual 37-139 Records Disposition Schedule
ASD(C^3I)	Assistant Secretary of Defense for Command, Control, Communications and Intelligence
AI	Artificial Intelligence
DASD(IM)	Deputy Assistant Secretary of Defense for Information Management
DLT	Decision Logic Table (Found in AFMAN 37-139)
DFD	Data Flow Diagram
DoD	Department of Defense
FOIA	Freedom of Information Act
MOC	Modified Overlap Coefficient
NLP	Natural Language Processing
RACS	Records Analysis and Classification System
RMA	Records Management Application
RM-BPR	DoD Records Management Business Process Reengineering
RMTF	DoD Records Management Task Force
SECAF	Secretary of the Air Force
USAF	United States Air Force

Appendix B: Overview of the RACS System

The RACS program is a proof of concept automated records analysis and classification system. The system takes as its input the metadata on a new record to be classified, processes that input, and based on that processing, presents the user with an ordered list of the most likely record classes to which the new record belongs. To support this basic functionality, RACS includes many administration functions which were implemented to manage the data files used by the system. The following sections briefly describe these functions and serve as a simple users manual for running RACS.

RACS Files

The RACS program requires several files to function properly. Additionally, files are created at runtime for various purposes. These files and their purposes are listed in Table 10.

Table 10. RACS Files

File Name	Purpose
racs.exe	RACS executable program (See Appendix C for the complete source code)
config.txt	Configuration file which racs.exe uses at run time (See Appendix D)
stoplist.txt	Stoplist used during the generation of record and class templates (See Appendix E)
cat1-cat5.dbf	Database files created to store the metadata for all records placed in a given record class (See Table 5 for the record classes which correspond to each database file)
cat1-cat5.dbb	Backup files created for each database file
cat1-cat5.tpl	Files containing the class templates for each database/record class
cat1tpl-cat5tpl.txt	Text versions of the five class templates (See Appendix H for a typical class template)
logfile.txt	A detailed log file containing details of each record classified (See Appendix F for an excerpt from logfile.txt)
scorelog.txt	A log file which records the correct database and offsets for each new record classified (See Appendix G for an excerpt from scorelog.txt)
logfile.bak	Backup file of logfile.txt.
scorelog.bak	Backup file of scorelog.txt.

RACS Menu/Interface Hierarchy

The RACS program presents the user with a series of menus and interfaces which control the execution of the program. Figure 18 illustrates the hierarchy of menus and user interfaces; for example, the Database Management Menu is a sub-menu of the Main Menu and the View/Edit Records Menu is a sub-menu of the Database Management Menu. The following sections describe the functions associated with each menu and interface.

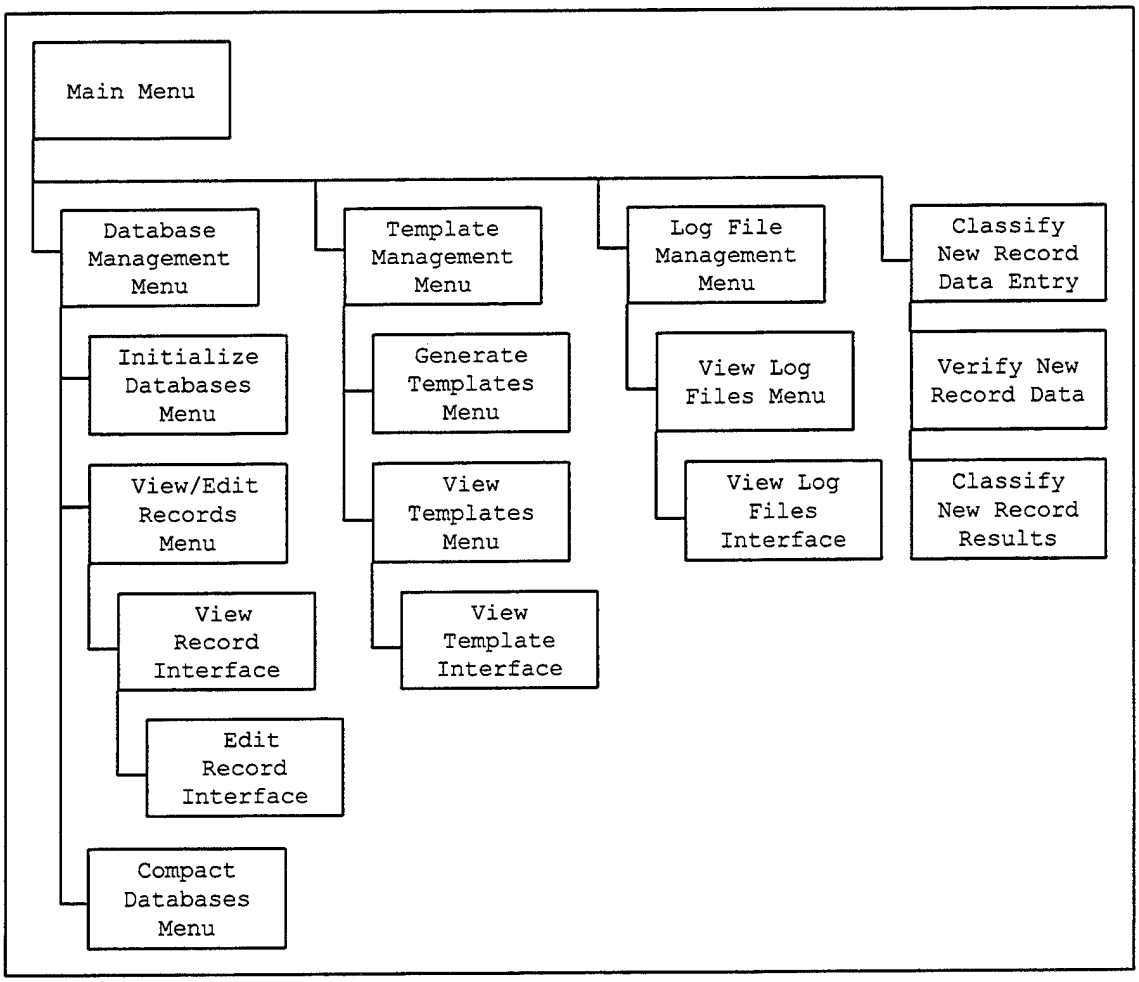

Figure 18. RACS Menu/Interface Hierarchy

Main Menu

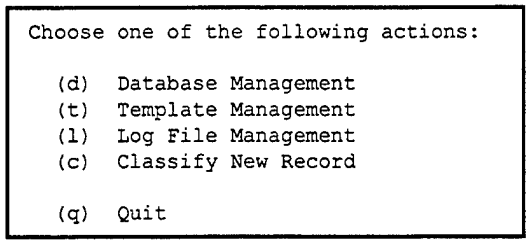

Figure 19. Main Menu

The options on the Main Menu perform the following functions:

(d) Opens the Database Management Menu.

(t) Opens the Template Management Menu.

(l) Opens the Log File Management Menu.

(c) Takes the user to the Classify New Record Data Entry interface for entering a new record to be classified. (See Chapter III for a detailed discussion of this process)

(q) Exits RACS.

Database Management Menu

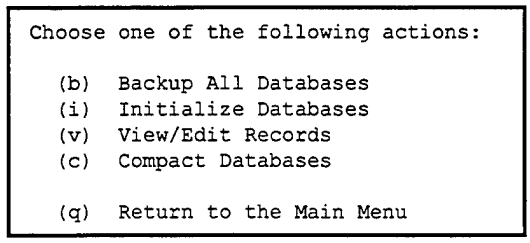

Figure 20. Database Management Menu

The options on the Database Management Menu perform the following functions:

(b) Creates backup copies of the five record class databases.

(i) Opens the Initialize Database Menu.

(v) Opens the View/Edit Records Menu.

(c) Opens the Compact Databases Menu.

(q) Returns the user to the Main Menu.

Initialize Databases Menu

```
Select the database to initialize:

  (1)  T  11-02 R 21 Item 3
  (2)  T  11-01 R 01 Item 6-3-2
  (3)  T  11-01 R 01 Item 6-4
  (4)  T  11-02 R 33 Item 12
  (5)  T 900-02 R 02 Item 15
  (a)  Initialize All Databases

  (q)  Return to the Databases Menu
```

Figure 21. Initialize Databases Menu

The options on the Initialize Databases Menu Perform the following functions:

(1-5) Initializes the selected database. Initializing a database deletes all records currently in the database and resets all of its internal values such as number of records to their initial values. Note: before a database is initialized RACS creates a backup copy of the database.

(a) Initializes all of the databases.

(q) Returns the user to the Database Menu.

View/Edit Records Menu

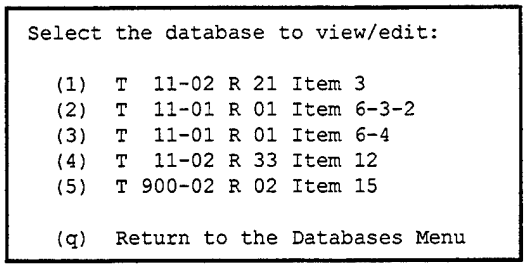

Figure 22. View/Edit Records Menu

The options on the View/Edit Records Menu perform the following functions.

(1-5) Opens the View Record Interface for the selected database/record class.

(q) Returns the user to the Database Menu.

View Record Interface

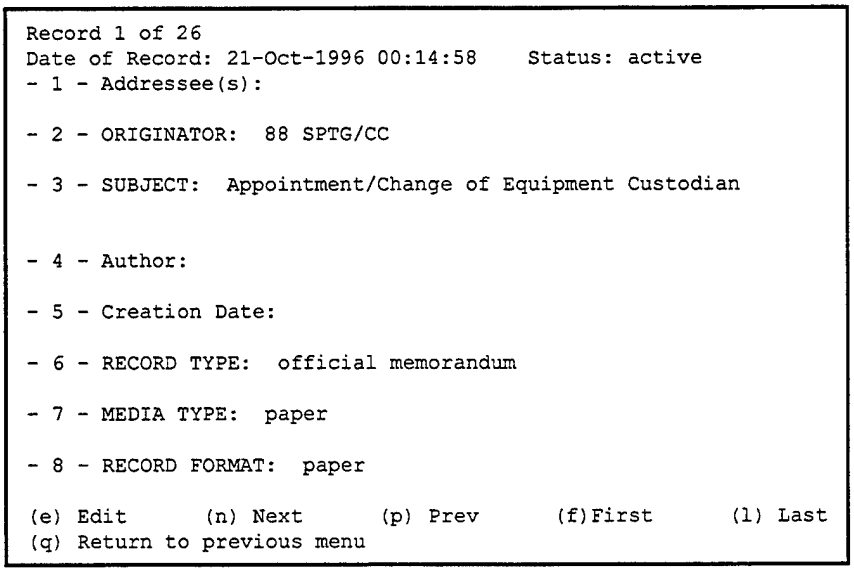

Figure 23. View Record Interface

The options presented on the View Record Interface perform the following functions:

(e) Opens the Edit Record Interface for the current record.

(n) Moves to the next record unless the user is currently viewing the last record.

(p) Moves to the previous record unless the user is currently viewing the first record.

(f) Moves to the first record in the database.

(l) Moves to the last record in the database.

(q) Returns the user to the View/Edit Records Menu.

Edit Record Interface

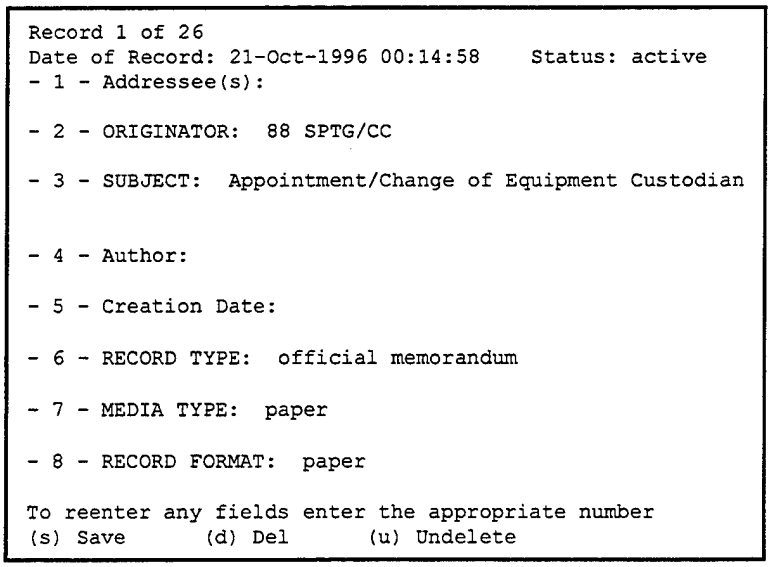

Figure 24. Edit Record Interface

The options presented on the Edit Record Interface perform the following functions:

(1-8) Allows user to reenter the data in the selected field.

(s) Saves the current record and returns to the View Record Interface. Even if no changes were made the user must select this option to exit this interface.

(d) Marks the current record as deleted. The Status field will change from "active" to "deleted."

(u) Marks the current record as active. The Status field will change from "deleted" to "active."

Compact Databases Menu

```
Select the database to compact:

  (1)  T  11-02 R 21 Item 3
  (2)  T  11-01 R 01 Item 6-3-2
  (3)  T  11-01 R 01 Item 6-4
  (4)  T  11-02 R 33 Item 12
  (5)  T 900-02 R 02 Item 15
  (a)  Compact All Databases

  (q)  Return to the Databases Menu
```

Figure 25. Compact Databases Menu

The options on the Compact Databases Menu perform the following functions:

(1-5) Compacts the selected database. Compacting a database rewrites the database, removing any records marked for deletion. Until a database is compacted, any records marked as "deleted" are still held in the database and can be undeleted from the Edit Record Interface.

(a) Compacts all five databases.

(q) Returns the user to the Database Menu.

Template Management Menu

```
Choose one of the following actions:

  (g)  Generate Templates
  (v)  View Templates

  (q)  Return to the Main Menu
```

Figure 26. Template Management Menu

The options on the Template Management Menu perform the following functions:

(g) Opens the Generate Templates Menu.

(v) Opens the View Templates Menu.

(q) Returns the user to the Main Menu.

Generate Templates Menu

```
Select the template to generate:

  (1)  T  11-02 R 21 Item 3
  (2)  T  11-01 R 01 Item 6-3-2
  (3)  T  11-01 R 01 Item 6-4
  (4)  T  11-02 R 33 Item 12
  (5)  T 900-02 R 02 Item 15
  (a)  Generate All Templates

  (q)  Return to the Templates Menu
```

Figure 27. Generate Templates Menu

The options on the Generate Templates Menu perform the following functions:

(1-5) Generates the class template for the selected database. The class template is the representation of a record class which RACS uses to determine the correct classification for a new record. (A complete discussion on the method for creating class templates is contained in Chapter III)

(a) Generates all five class templates for the five databases/record class.

(q) Returns the user to the Template Management Menu.

View Templates Menu

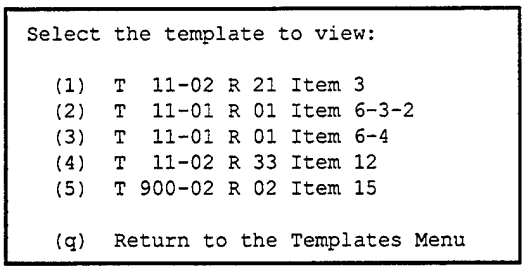

Figure 28. View Templates Menu

The options on the View Templates Menu perform the following functions:

(1-5) Opens the selected class template for viewing with the View Template Interface.

(q) Returns the user to the Template Management Menu.

View Template Interface

The View Template Interface is simply the MS-DOS® edit utility. The selected class template is automatically opened for viewing. Once done viewing the template the user exits by pressing ALT-F then X.

Log Files Management Menu

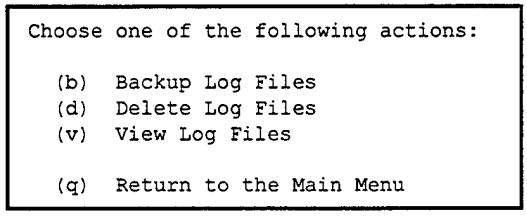

Figure 29. Log Files Management Menu

The options on the Log Files Management Menu perform the following functions:

(b) Creates backup copies of both the logfile.txt and scorelog.txt log files.

(d) Creates backup copies of both log files and then deletes the original copies.

(v) Opens the View Log Files Menu.

(q) Returns the user to the Main Menu.

View Log Files Menu

```
Select the log file to view:

  (a)  All Details
  (s)  Only Score

  (q)  Return to the Log Files Menu
```

Figure 30. View Log Files Menu

The options on the View Log Files Menu perform the following functions:

(a) Opens the log file logfile.txt in the View Log Files Interface.

(s) Opens the log file scorelog.txt in the View Log Files Interface.

(q) Returns the user to the Log Files Menu.

View Log Files Interface

The View Log Files Interface is simply the MS-DOS® edit utility. The selected log file is automatically opened for viewing. Once done viewing the log file the user exits by pressing ALT-F then X.

Classify New Record Data Entry

```
Addressee(s):
ORIGINATING ORGANIZATION:  ASC/CVH
SUBJECT:  Focal Points for Management Operations
Author:
Creation Date:
RECORD TYPE:  official memorandum
MEDIA TYPE:  paper
RECORD FORMAT:  paper
```

Figure 31. Classify New Record Data Entry

The field names appear one at a time for the user to enter data. To proceed to the next field the user presses the Enter key. The field names in all capital letters indicate the fields which are actually used in the classification process. The Verify New Record Data interface is opened when the user presses the enter key after entering data in the RECORD FORMAT field.

Verify New Record Data

```
Date of Record: 23-Oct-1996 17:57:30

- 1 - Addressee(s):

- 2 - ORIGINATOR:  ASC/CVH

- 3 - SUBJECT:  Focal Points for Management Operations


- 4 - Author:

- 5 - Creation Date:

- 6 - RECORD TYPE:  official memorandum

- 7 - MEDIA TYPE:  paper

- 8 - RECORD FORMAT:  paper

To reenter any fields enter the appropriate number
(a) to accept and process the record
```

Figure 32. Verify New Record Data

The options presented on the Verify New Record Data interface perform the following functions:

(1-8) Allows user to reenter the data in the selected field.

(a) Accepts the new data entered and causes RACS to evaluate the new record to determine its classification. (See Chapter III for a complete discussion of this process)

Classify New Record Results

```
Select the correct database:
30/20/30/10/10        20/20/20/20/20        50/30/00/10/10
DBF RANK SCORE        DBF RANK SCORE        DBF RANK SCORE
 4   1   0.550         4   1   0.638         5   1   0.321
 1   2   0.496         1   2   0.604         4   2   0.319
 2   3   0.463         2   3   0.580         1   3   0.267
 5   4   0.422         5   4   0.572         2   4   0.237
 3   5   0.200         3   5   0.400         3   5   0.200

1  T  11-02 R 21 Item 3
   Delegations/Designations of Authority & Additional Duty Assignments
2  T  11-01 R 01 Item 6-3-2
   Office Administrative Files - Internal Administration or Housekeeping
   -- Supplies/Equipment
3  T  11-01 R 01 Item 6-4
   Office Administrative Files - Internal Administration or Housekeeping
   -- Safety
4  T  11-02 R 33 Item 12
   Internal Inspections/Self-Inspection Check Lists/Inventories
5  T 900-02 R02 Item 15
   Suggestions, Inventions, & Scientific Achievements - at Evaluation Office
```

Figure 33. Classify New Record Results

This interface presents the results of RACS' analysis of the new record data. The user enters the number corresponding to the correct database/record class for the new record.

Appendix C: RACS Source Code

```
/*-----------------------------------------------------------------------------
PROJECT: racs.prj

FILE: racs.h

PURPOSE:
  This is the single header file included by every module.
-----------------------------------------------------------------------------*/

#include "includes.h"
#include "variable.h"
#include "defines.h"
#include "prototyp.h"
```

```
/*-------------------------------------------------------------------------
PROJECT: racs.prj

FILE: includes.h

PURPOSE:
  This file lists all standard header files required by racs.exe
-------------------------------------------------------------------------*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include <conio.h>
#include <time.h>
#include <ctype.h>
```

```
/*-----------------------------------------------------------------------------
PROJECT: racs.prj

FILE: variable.h

PURPOSE:
  This file defines all global variables and structures.
-----------------------------------------------------------------------------*/

#ifndef VARIABLE_H
#define VARIABLE_H

#ifndef EXTERN
#define EXTERN
#endif

/* Structure for standard dBase III file header */
EXTERN struct DB3HEADER {
   unsigned int bfVersion:7;
   unsigned int bfHasMemo:1;
   unsigned int bYear:8;
   unsigned char bMonth;
   unsigned char bDay;
   long int lNumberRecords;
   short int nFirstRecordOffset;
   short int nRecordLength;
   unsigned char szReserved[20];
};

/* Structure for standard dBase III column headers */
EXTERN struct COLUMNDEF {
   char szColumnName[11];
   char chType;
   long lFieldPointer;
   unsigned char byLength;
   unsigned char byDecimalPlace;
   char szReserved[14];
};

/* Structure for individual record data */
EXTERN struct DB3RECORD {
   char szStatus[1]; /* does not count as member */
   char szDateRecord[26];
   char szTo[101];
   char szOriginOrg[101];
   char szSubject[255];
   char szAuthor[101];
   char szCreateDate[26];
   char szRecType[51];
   char szMediaType[51];
   char szRecFormat[51];
};

/* Structure to hold keywords with their frequencies */
EXTERN struct KEY {
   int iFreq;
   char szKwrd[21];
};

/* Structure to hold record template information */
EXTERN struct RECTMPLT {
   struct KEY pSub[30];
   struct KEY pOrg[10];
   struct KEY pTyp[5];
   struct KEY pMed[5];
```

```
   struct KEY pFrm[5];
};

/* Structure to hold class template information */
EXTERN struct CLASSTMPLT {
   int iNumRecs;
   struct KEY pSub[1000];
   struct KEY pOrg[100];
   struct KEY pTyp[30];
   struct KEY pMed[20];
   struct KEY pFrm[30];
};

/* Structures to hold the results of computing a MOC for each database */
EXTERN struct MOC {
   float fResult;
   float fTop;
   float fBottom;
   int iNumRecs;
};

EXTERN struct SCORE {
   int iDBFNum;
   int iRank[3];
   float fScore[3];
   struct MOC sub;
   struct MOC org;
   struct MOC typ;
   struct MOC med;
   struct MOC frm;
};

/* Structure to hold raw data for each term analyzed */
EXTERN struct TERM {
   char szTerm[51];
   char szToken[41];
   int iTokenType;
};

/* Structure to hold raw text analysis information */
EXTERN struct ANALYSIS {
   int iNumTerms[5];
   struct TERM term[50];
};

/* Token Types for getTerms() in analyzer.c */
enum
{
   LEXERROR,
   ALLALPHA,
   NONWORD,
   PUNCT,
   EOL,
   UNDEFINED
};

#endif
```

```
/*-------------------------------------------------------------------------
PROJECT: racs.prj

FILE: defines.h

PURPOSE:
  All global defines are listed in this header.
-------------------------------------------------------------------------*/

#ifndef DEFINES_H
#define DEFINES_H

#define TRUE 1
#define FALSE 0

/* The following defines are used within the db3funct.c module */
#define DELETED_RECORD    '*'
#define USABLE_RECORD     ' '

#define NUMERIC_FIELD     'N'
#define CHARACTER_FIELD   'C'
#define LOGICAL_FIELD     'L'
#define MEMO_FIELD        'M'
#define DATE_FIELD        'D'
#define FLOAT_FIELD       'F'
#define PICTURE_FIELD     'P'

#endif
```

```
/*-------------------------------------------------------------------------
PROJECT: racs.prj

FILE: prototyp.h

PURPOSE:
  All function prototypes are included here with reference to the
  module where the given function is defined.
-------------------------------------------------------------------------*/

#ifndef PROTOTYP_H
#define PROTOTYP_H

/* analyzer.c */
void getTerms(char **ppText, char *pToken, int *pTokenType);
char *numToken(int iTokenType);

/* classrec.c */
void classifyRecord(void);
void getNewRecord(struct DB3RECORD *pdb3record);
void genRECTMPLT(struct DB3RECORD *pdb3record, struct RECTMPLT *pRecTmplt,
   struct ANALYSIS *pAnalysis);
void addToRECTMPLT(char *szTerm, struct RECTMPLT *pCurRec, int iTMPLTfield);
int chooseDBF(struct SCORE *pScore);
void logRECTMPLT(struct ANALYSIS *pAnalysis, struct RECTMPLT *pCurRec,
   struct SCORE *pScore, int iDBFNum);

/* db3funct.c */
void createDBF(int iDBFNum);
void addRecord(int iDBFNum, struct DB3RECORD *pdb3record);
void displayRecords(int iDBFNum);
void editRecord(struct DB3RECORD *pdb3record, struct DB3HEADER *pdb3header,
   int iRecNum);
void compactDBF(int iDBFNum);

/* main.c */
void main(void);
char *szpGetConfig(char szHeadText[], int iFileNum);
void displayError(char szErrorMessage[]);
void copyFile(char *oldName, char *newName);
void cleanUp(void);

/* menus.c */
void introScreen(void);
void mainMenu(void);
void databaseMenu(void);
void initializeDatabaseMenu(void);
void viewDatabaseMenu(void);
void compactDatabaseMenu(void);
void initializeDBF(int iDBFNum);
void templateMenu(void);
void generateTemplatesMenu(void);
void viewTemplatesMenu(void);
void logFileMenu(void);
void viewLogFileMenu(void);

/* score.c */
void compareTemplates(struct RECTMPLT *pRecTmplt, struct SCORE *pScore);
void calcScore(struct RECTMPLT *pRecTmplt, struct CLASSTMPLT *pClsTmplt,
   struct SCORE *pScore);

/* stemmer.c */
char *stem(register char *word);
static int WordSize(register char *word);
static int ContainsVowel(register char *word);
```

```
static int EndsWithCVC(register char *word);
static int AddAnE(register char *word);
static int RemoveAnE(register char *word);
static int ReplaceEnd(register char *word, struct RULELIST *rule);

/* stoplist.c */
int loadStoplist(char *szStoplist[]);
void unloadStoplist(char *szStoplist[], int iNumWords);
int checkStoplist(char *szTerm, char *szStoplist[], int iNumWords);

/* template.c */
int genCLASSTMPLT(int iDBFNum);
void addToCLASSTMPLT(char *szTerm, struct CLASSTMPLT *pTmplt, int iField);
void logCLASSTMPLT(struct CLASSTMPLT *pTmplt, int iDBFNum);

#endif
```

```
/*-----------------------------------------------------------------------
PROJECT: racs.prj

FILE: analyzer.c

PURPOSE:
  This module contains the functions which perform lexical analysis of the
  individual terms extratcted from various record metadata fields.

FUNCTIONS:
  void getTerms(char **ppText, char *pToken, int *pTokenType)
  char *numToken(int iTokenType)
-----------------------------------------------------------------------*/

#define EXTERN extern
#include "racs.h"


void getTerms(char **ppText, char *pToken, int *pTokenType)
{
   for( ; **ppText == ' ' || **ppText == '\t'; (*ppText)++);

   if(**ppText == '\0')
   {
      *pTokenType = EOL;
      return;
   }

   if((**ppText >= 'A' && **ppText <= 'Z') ||
      (**ppText >= 'a' && **ppText <= 'z'))
   {
      *pToken++ = *(*ppText)++;
      if(**ppText >= 'a' && **ppText <= 'z')
      {
         while(**ppText >= 'a' && **ppText <= 'z')
         {
            *pToken++ = *(*ppText)++;
         }
      *pToken = '\0';
      *pTokenType = ALLALPHA;
      return;
      }
      *pToken-- = *(*ppText)--;
   }

   if(**ppText >= 'A' && **ppText <= 'Z')
   {
      *pToken++ = *(*ppText)++;
      if(**ppText >= 'A' && **ppText <= 'Z')
      {
         while((**ppText >= 'A' && **ppText <= 'Z') ||
               (**ppText >= 'a' && **ppText <= 'z'))
         {
            *pToken++ = *(*ppText)++;
         }
      *pToken = '\0';
      *pTokenType = NONWORD;
      return;
      }
      *pToken-- = *(*ppText)--;
   }

   if(**ppText >= '1' && **ppText <= '9')
   {
      *pToken++ = *(*ppText)++;
```

```
      if((**ppText >= '0' && **ppText <= '9') ||
         (**ppText == '/') || (**ppText == '-'))
      {
         while((**ppText >= '0' && **ppText <= '9') ||
               (**ppText == '/') || (**ppText == '-'))
         {
            *pToken++ = *(*ppText)++;
         }
      *pToken = '\0';
      *pTokenType = NONWORD;
      return;
      }
      *pToken-- = *(*ppText)--;
   }

   if((**ppText >= 'A' && **ppText <= 'Z') ||
      (**ppText >= 'a' && **ppText <= 'z'))
   {
      *pToken++ = *(*ppText)++;
      *pToken = '\0';
      *pTokenType = ALLALPHA;
      return;
   }

   if(**ppText >= '0' && **ppText <= '9')
   {
      *pToken++ = *(*ppText)++;
      *pToken = '\0';
      *pTokenType = NONWORD;
      return;
   }

   if((**ppText >= 33 && **ppText <= 47) ||
      (**ppText >= 58 && **ppText <= 64) ||
      (**ppText >= 91 && **ppText <= 96) ||
      (**ppText >= 123 && **ppText <= 126))
   {
      *pToken++ = *(*ppText)++;
      *pToken = '\0';
      *pTokenType = PUNCT;
      return;
   }

   *pTokenType = LEXERROR;
   return;
}


char *numToken(int iTokenType)
{
   static char *tokenNum[] =
   {
      "LEXERROR",
      "ALLALPHA",
      "NONWORD",
      "PUNCT",
      "EOL",
      "UNDEFINED"
   };
   return(tokenNum[iTokenType]);
}
```

```
/*-----------------------------------------------------------------------------
PROJECT: racs.prj

FILE: classrec.c

PURPOSE:
  The functions in this module are the heart of the RACS program.
  The function classifyRecord() controls the actual process of accepting
  and classifying a new record.

FUNCTIONS:
  void classifyRecord(void)
  void getNewRecord(struct DB3RECORD *pdb3record)
  void genRECTMPLT(struct DB3RECORD *pdb3record, struct RECTMPLT *pRecTmplt,
     struct ANALYSIS *pAnalysis)
  void addToRECTMPLT(char *szTerm, struct RECTMPLT *pCurRec, int iTMPLTfield)
  int chooseDBF(struct SCORE *pScore)
  void logRECTMPLT(struct ANALYSIS *pAnalysis, struct RECTMPLT *pCurRec
     struct SCORE *pScore, int iDBFNum)
-----------------------------------------------------------------------------*/

#define EXTERN extern
#include "racs.h"


void classifyRecord(void)
{
   struct DB3RECORD db3record;
   struct DB3RECORD *pdb3record;

   struct RECTMPLT recTmplt;
   struct RECTMPLT *pRecTmplt;

   struct ANALYSIS analysis;
   struct ANALYSIS *pAnalysis;

   struct SCORE score[5];
   struct SCORE *pScore;

   int iDBFNum;

   int i;

   pdb3record = &db3record;
   pRecTmplt = &recTmplt;
   pAnalysis = &analysis;
   pScore = &score[0];

   memset(&recTmplt, 0, sizeof(struct RECTMPLT));
   memset(&db3record, 0, sizeof(struct DB3RECORD));
   memset(&analysis, 0, sizeof(struct ANALYSIS));
   memset(&score, 0, sizeof(struct SCORE));

   getNewRecord(pdb3record);

   genRECTMPLT(pdb3record, pRecTmplt, pAnalysis);

   compareTemplates(pRecTmplt, pScore);

   iDBFNum = chooseDBF(pScore);

   logRECTMPLT(pAnalysis, pRecTmplt, pScore, iDBFNum);

   addRecord(iDBFNum, pdb3record);
```

```
   genCLASSTMPLT(iDBFNum);
}


void getNewRecord(struct DB3RECORD *pdb3record)
{
   int iDone = FALSE;
   int iFirstTime = TRUE;

   char cSel;
   char szBuff[255];

   struct tm *curTime;
   time_t tClock;

   clrscr();

   pdb3record->szStatus[0] = USABLE_RECORD;

   time(&tClock);
   curTime = localtime(&tClock);
   strftime(szBuff, 255, "%d-%b-%Y %X", curTime);
   strncpy(pdb3record->szDateRecord, szBuff,
        sizeof(pdb3record->szDateRecord));

   cSel = 48;
   while(iDone == FALSE)
   {
      _setcursortype(_NORMALCURSOR);
      if(iFirstTime == TRUE)
      {
         cSel++;
      }

      switch(cSel)
      {
         case 49:
            printf("Addressee(s): ");
            gets(szBuff);
            strncpy(pdb3record->szTo, szBuff,
                 sizeof(pdb3record->szTo));
            if(iFirstTime == FALSE)
            {
               cSel = 0;
            }
            break;

         case 50:
            printf("ORIGINATING ORGANIZATION: ");
            gets(szBuff);
            strncpy(pdb3record->szOriginOrg, szBuff,
                 sizeof(pdb3record->szOriginOrg));
            if(iFirstTime == FALSE)
            {
               cSel = 0;
            }
            break;

         case 51:
            printf("SUBJECT: ");
            gets(szBuff);
            strncpy(pdb3record->szSubject, szBuff,
                 sizeof(pdb3record->szSubject));
            if(iFirstTime == FALSE)
            {
```

```
      cSel = 0;
   }
   break;

case 52:
   printf("Author: ");
   gets(szBuff);
   strncpy(pdb3record->szAuthor, szBuff,
        sizeof(pdb3record->szAuthor));
   if(iFirstTime == FALSE)
   {
      cSel = 0;
   }
   break;

case 53:
   printf("Creation Date: ");
   gets(szBuff);
   strncpy(pdb3record->szCreateDate, szBuff,
        sizeof(pdb3record->szCreateDate));
   if(iFirstTime == FALSE)
   {
      cSel = 0;
   }
   break;

case 54:
   printf("RECORD TYPE: ");
   gets(szBuff);
   strncpy(pdb3record->szRecType, szBuff,
        sizeof(pdb3record->szRecType));
   if(iFirstTime == FALSE)
   {
      cSel = 0;
   }
   break;

case 55:
   printf("MEDIA TYPE: ");
   gets(szBuff);
   strncpy(pdb3record->szMediaType, szBuff,
        sizeof(pdb3record->szMediaType));
   if(iFirstTime == FALSE)
   {
      cSel = 0;
   }
   break;

case 56:
   printf("RECORD FORMAT: ");
   gets(szBuff);
   strncpy(pdb3record->szRecFormat, szBuff,
        sizeof(pdb3record->szRecFormat));
   iFirstTime = FALSE;
   cSel = 0;
   if(iFirstTime == FALSE)
   {
      cSel = 0;
   }
   break;

case 'a':
case 'A':
   iDone = TRUE;
   break;
```

```
            default:
               clrscr();
               _setcursortype(_NOCURSOR);
               printf("Date of Record:  %s\n\n",
                     pdb3record->szDateRecord);
               printf("- 1 - Addressee(s):  %-100s\n",
                     pdb3record->szTo);
               printf("- 2 - ORIGINATOR:  %-100s\n",
                     pdb3record->szOriginOrg);
               printf("- 3 - SUBJECT:  %-254s\n",
                     pdb3record->szSubject);
               printf("- 4 - Author:  %-100s\n",
                     pdb3record->szAuthor);
               printf("- 5 - Creation Date:  %-25s\n\n",
                     pdb3record->szCreateDate);
               printf("- 6 - RECORD TYPE:  %-50s\n\n",
                     pdb3record->szRecType);
               printf("- 7 - MEDIA TYPE:  %-50s\n\n",
                     pdb3record->szMediaType);
               printf("- 8 - RECORD FORMAT:  %-50s\n\n",
                     pdb3record->szRecFormat);
               puts("\nTo reenter any fields enter the appropriate number");
               puts("(a) to accept and process the record\n");
               cSel = getch();
         }
      }
      _setcursortype(_NOCURSOR);
   }

   void genRECTMPLT(struct DB3RECORD *pdb3record, struct RECTMPLT *pRecTmplt,
               struct ANALYSIS *pAnalysis)
   {
      char szBuff[255];
      char szTermBuff[51];
      char *pString, *pToken;
      char szToken[41];
      int iTokenType;
      int iRecField;

      char *szStoplist[500];
      int iNumWords;
      int iMatch;

      int i, j, k;

      iNumWords = loadStoplist(szStoplist);

      i = 0;
      for(j = 0; j < 5; j++)
      {
         switch(j)
         {
            case 0:
               strcpy(szBuff, pdb3record->szSubject);
               iRecField = 's';
               break;

            case 1:
               strcpy(szBuff, pdb3record->szOriginOrg);
               iRecField = 'o';
               break;

            case 2:
```

```
      strcpy(szBuff, pdb3record->szRecType);
      iRecField = 't';
      break;

   case 3:
      strcpy(szBuff, pdb3record->szMediaType);
      iRecField = 'm';
      break;

   case 4:
      strcpy(szBuff, pdb3record->szRecFormat);
      iRecField = 'f';
      break;
}

pString = szBuff;

iTokenType = UNDEFINED;

k = 1;
while(iTokenType != EOL && iTokenType != LEXERROR)
{
   pToken = szToken;

   getTerms(&pString, pToken, &iTokenType);

   if(iTokenType != EOL)
   {
      pAnalysis->term[i].iTokenType = iTokenType;
      strcpy(pAnalysis->term[i].szToken, szToken);

      if(iTokenType == ALLALPHA)
      {
         strcpy(szTermBuff, strlwr(szToken));

         iMatch = checkStoplist(szTermBuff, szStoplist, iNumWords);

         if(iMatch == TRUE)
         {
            strcpy(szTermBuff, "-SW-");
         }
         else
         {
            strcpy(szTermBuff, stem(strlwr(szToken)));
            addToRECTMPLT(szTermBuff, pRecTmplt, iRecField);
         }
      }

      if(iTokenType == NONWORD)
      {
         strcpy(szTermBuff, strlwr(szToken));

         iMatch = checkStoplist(szTermBuff, szStoplist, iNumWords);

         if(iMatch == TRUE)
         {
            strcpy(szTermBuff, "-SW-");
         }
         else
         {
            strcpy(szTermBuff, strlwr(szToken));
            addToRECTMPLT(szTermBuff, pRecTmplt, iRecField);
         }
      }
```

```
            if(iTokenType == PUNCT)
            {
               strcpy(szTermBuff, "----");
            }
            strcpy(pAnalysis->term[i].szTerm, szTermBuff);
            pAnalysis->iNumTerms[j] = k;
            i++;
            k++;
         }
      }
   }
   unloadStoplist(szStoplist, iNumWords);
}


void addToRECTMPLT(char *szTerm, struct RECTMPLT *pCurRec, int iTMPLTfield)
{
   int i;

   if(iTMPLTfield == 's')
   {
      for(i = 0; pCurRec->pSub[i].szKwrd[0] != '\0'; i++)
      {
         if(strcmp(pCurRec->pSub[i].szKwrd, szTerm) == 0)
         {
            pCurRec->pSub[i].iFreq += 1;
            return;
         }
      }
      strcpy(pCurRec->pSub[i].szKwrd, szTerm);
      pCurRec->pSub[i].iFreq = 1;
   }

   if(iTMPLTfield == 'o')
   {
      for(i = 0; pCurRec->pOrg[i].szKwrd[0] != '\0'; i++)
      {
         if(strcmp(pCurRec->pOrg[i].szKwrd, szTerm) == 0)
         {
            pCurRec->pOrg[i].iFreq += 1;
            return;
         }
      }
      strcpy(pCurRec->pOrg[i].szKwrd, szTerm);
      pCurRec->pOrg[i].iFreq = 1;
   }

   if(iTMPLTfield == 't')
   {
      for(i = 0; pCurRec->pTyp[i].szKwrd[0] != '\0'; i++)
      {
         if(strcmp(pCurRec->pTyp[i].szKwrd, szTerm) == 0)
         {
            pCurRec->pTyp[i].iFreq += 1;
            return;
         }
      }
      strcpy(pCurRec->pTyp[i].szKwrd, szTerm);
      pCurRec->pTyp[i].iFreq = 1;
   }

   if(iTMPLTfield == 'm')
   {
      for(i = 0; pCurRec->pMed[i].szKwrd[0] != '\0'; i++)
      {
```

```
            if(strcmp(pCurRec->pMed[i].szKwrd, szTerm) == 0)
            {
               pCurRec->pMed[i].iFreq += 1;
               return;
            }
         }
         strcpy(pCurRec->pMed[i].szKwrd, szTerm);
         pCurRec->pMed[i].iFreq = 1;
      }

      if(iTMPLTfield == 'f')
      {
         for(i = 0; pCurRec->pFrm[i].szKwrd[0] != '\0'; i++)
         {
            if(strcmp(pCurRec->pFrm[i].szKwrd, szTerm) == 0)
            {
               pCurRec->pFrm[i].iFreq += 1;
               return;
            }
         }
         strcpy(pCurRec->pFrm[i].szKwrd, szTerm);
         pCurRec->pFrm[i].iFreq = 1;
      }
   }


   int chooseDBF(struct SCORE *pScore)
   {
      int i, j;
      int iInner, iOuter;
      int iDBFNum;
      int iDone = FALSE;
      int iDuplicate[3];
      char cSel;

      struct RANK {
         int iDBFNum;
         int iRank;
         float fScore;
         struct MOC sub;
         struct MOC org;
         struct MOC typ;
         struct MOC med;
         struct MOC frm;
      };

      struct RANK rank[5][3];
      struct RANK tempRank;

      /* Transfer all values to the structure rank for ease of manipulation  */
      for(i = 0; i < 5; i++)
      {
         for(j = 0; j < 3; j++)
         {
            rank[i][j].iDBFNum = pScore->iDBFNum;
            rank[i][j].fScore = pScore->fScore[j];
            rank[i][j].iRank = 1;
            rank[i][j].sub = pScore->sub;
            rank[i][j].org = pScore->org;
            rank[i][j].typ = pScore->typ;
            rank[i][j].med = pScore->med;
            rank[i][j].frm = pScore->frm;
         }
         pScore++;
      }
```

```
for(i = 0; i < 5; i++)
{
   pScore--;
}

/* Order the scores for presentation to the user using a bubble sort */
for(i = 0; i < 3; i++)
{
   for(iOuter = 0; iOuter < 4; iOuter++)
   {
      for(iInner = iOuter; iInner < 5; iInner++)
      {
         if(rank[iInner][i].fScore > rank[iOuter][i].fScore)
         {
            tempRank.iDBFNum = rank[iInner][i].iDBFNum;
            tempRank.fScore = rank[iInner][i].fScore;
            tempRank.sub = rank[iInner][i].sub;
            tempRank.org = rank[iInner][i].org;
            tempRank.typ = rank[iInner][i].typ;
            tempRank.med = rank[iInner][i].med;
            tempRank.frm = rank[iInner][i].frm;
            rank[iInner][i].iDBFNum = rank[iOuter][i].iDBFNum;
            rank[iInner][i].fScore = rank[iOuter][i].fScore;
            rank[iInner][i].sub = rank[iOuter][i].sub;
            rank[iInner][i].org = rank[iOuter][i].org;
            rank[iInner][i].typ = rank[iOuter][i].typ;
            rank[iInner][i].med = rank[iOuter][i].med;
            rank[iInner][i].frm = rank[iOuter][i].frm;
            rank[iOuter][i].iDBFNum = tempRank.iDBFNum;
            rank[iOuter][i].fScore = tempRank.fScore;
            rank[iOuter][i].sub = tempRank.sub;
            rank[iOuter][i].org = tempRank.org;
            rank[iOuter][i].typ = tempRank.typ;
            rank[iOuter][i].med = tempRank.med;
            rank[iOuter][i].frm = tempRank.frm;
         }
      }
   }
}

for(i = 0; i < 3; i++)
{
   iDuplicate[i] = 0;
   rank[0][i].iRank = 1;
   for(j = 1; j < 5; j++)
   {
      if(rank[j][i].fScore == rank[j - 1][i].fScore)
      {
         rank[j][i].iRank = rank[j - 1][i].iRank;
         iDuplicate[i] += 1;
      }
      else
      {
         rank[j][i].iRank = ((rank[j - 1][i].iRank) + 1 +
            iDuplicate[i]);
         iDuplicate[i] = 0;
      }
   }
}

while(iDone == FALSE)
{
   _setcursortype(_NOCURSOR);
   clrscr();
```

```
puts("Select the correct database:");
puts("30/20/30/10/10        20/20/20/20/20        50/30/00/10/10");
puts("DBF RANK SCORE        DBF RANK SCORE        DBF RANK SCORE");

for(i = 0; i < 5; i++)
{
   for(j = 0; j < 3; j++)
   {
      printf(" %d   %d   %.3f        ", rank[i][j].iDBFNum,
         rank[i][j].iRank, rank[i][j].fScore);
   }
   printf("\n");
}
printf("\n");
printf("1  T  11-02 R 21 Item 3\n");
printf("   Delegation/Designations of Authority &");
printf(" Additional Duty Assignments\n");

printf("2  T  11-01 R 01 Item 6-3-2\n");
printf("   Office Administrative Files - Internal");
printf(" Administration or Housekeeping\n");
printf("   -- Supplies/Eqiupment\n");

printf("3  T  11-01 R 01 Item 6-4\n");
printf("   Office Administrative Files - Internal");
printf(" Administration or Housekeeping\n");
printf("   -- Safety\n");

printf("4  T  11-02 R 33 Item 12\n");
printf("   Internal Inspections/Self-Inspection");
printf(" Check Lists/Inventories\n");

printf("5  T 900-02 R 02 Item 15\n");
printf("   Suggestions, Inventions, & Scientific");
printf(" Achievements - at Evaluation Office\n");

cSel = getch();

switch(cSel)
{
   case '1':
      iDBFNum = 1;
      iDone = TRUE;
      break;

   case '2':
      iDBFNum = 2;
      iDone = TRUE;
      break;

   case '3':
      iDBFNum = 3;
      iDone = TRUE;
      break;

   case '4':
      iDBFNum = 4;
      iDone = TRUE;
      break;

   case '5':
      iDBFNum = 5;
      iDone = TRUE;
      break;
```

```
            default:
               puts("\n  INVALID KEY!");
               delay(1000);
         }
      }

      /* Reorder the scores by database number */
      for(i = 0; i < 3; i++)
      {
         for(iOuter = 0; iOuter < 4; iOuter++)
         {
            for(iInner = iOuter; iInner < 5; iInner++)
            {
               if(rank[iInner][i].iDBFNum <= rank[iOuter][i].iDBFNum)
               {
                  tempRank.iDBFNum = rank[iInner][i].iDBFNum;
                  tempRank.fScore = rank[iInner][i].fScore;
                  tempRank.iRank = rank[iInner][i].iRank;
                  tempRank.sub = rank[iInner][i].sub;
                  tempRank.org = rank[iInner][i].org;
                  tempRank.typ = rank[iInner][i].typ;
                  tempRank.med = rank[iInner][i].med;
                  tempRank.frm = rank[iInner][i].frm;
                  rank[iInner][i].iDBFNum = rank[iOuter][i].iDBFNum;
                  rank[iInner][i].fScore = rank[iOuter][i].fScore;
                  rank[iInner][i].iRank = rank[iOuter][i].iRank;
                  rank[iInner][i].sub = rank[iOuter][i].sub;
                  rank[iInner][i].org = rank[iOuter][i].org;
                  rank[iInner][i].typ = rank[iOuter][i].typ;
                  rank[iInner][i].med = rank[iOuter][i].med;
                  rank[iInner][i].frm = rank[iOuter][i].frm;
                  rank[iOuter][i].iDBFNum = tempRank.iDBFNum;
                  rank[iOuter][i].fScore = tempRank.fScore;
                  rank[iOuter][i].iRank = tempRank.iRank;
                  rank[iOuter][i].sub = tempRank.sub;
                  rank[iOuter][i].org = tempRank.org;
                  rank[iOuter][i].typ = tempRank.typ;
                  rank[iOuter][i].med = tempRank.med;
                  rank[iOuter][i].frm = tempRank.frm;
               }
            }
         }
      }

      for(i = 0; i < 5; i++)
      {
         for(j = 0; j < 3; j++)
         {
            pScore->iDBFNum = rank[i][j].iDBFNum;
            pScore->iRank[j] = rank[i][j].iRank;
            pScore->fScore[j] = rank[i][j].fScore;
            pScore->sub = rank[i][j].sub;
            pScore->org = rank[i][j].org;
            pScore->typ = rank[i][j].typ;
            pScore->med = rank[i][j].med;
            pScore->frm = rank[i][j].frm;
         }
         pScore++;
      }
      return iDBFNum;
}


void logRECTMPLT(struct ANALYSIS *pAnalysis, struct RECTMPLT *pCurRec,
            struct SCORE *pScore, int iDBFNum)
```

```
{
   FILE *fpLogFile;
   FILE *fpScoreLog;
   char szHeader[20] = "logfile";

   struct SCORE *score;

   int iRank[3];
   int iDuplicate[3];
   int i, j, k;

   if((fpLogFile = fopen(szpGetConfig(szHeader, 1), "a")) == NULL)
   {
      displayError("opening log file");
   }

   if((fpScoreLog = fopen(szpGetConfig(szHeader, 2), "a")) == NULL)
   {
      displayError("opening log file");
   }

   for(i = 0; i < 38; i++)
   {
      fprintf(fpLogFile, "/\\");
   }
   fprintf(fpLogFile, "\n*****************************");
   fprintf(fpLogFile, " INPUT ANALYSIS ");
   fprintf(fpLogFile, "*****************************\n");


   i = 0;
   for(j = 0; j < 5; j++)
   {
      if(j == 0)
      {
         fprintf(fpLogFile, "SUBJECT:\n");
         for(k = 0; k < pAnalysis->iNumTerms[j]; k++)
         {
            fprintf(fpLogFile, "%-20s%-12s%s\n",
                  pAnalysis->term[i].szToken,
                  numToken(pAnalysis->term[i].iTokenType),
                  pAnalysis->term[i].szTerm);
            i++;
         }
         fprintf(fpLogFile, "\n");
      }

      if(j == 1)
      {
         fprintf(fpLogFile, "ORIGINATING ORGANIZATION:\n");
         for(k = 0; k < pAnalysis->iNumTerms[j]; k++)
         {
            fprintf(fpLogFile, "%-20s%-12s%s\n",
                  pAnalysis->term[i].szToken,
                  numToken(pAnalysis->term[i].iTokenType),
                  pAnalysis->term[i].szTerm);
            i++;
         }
         fprintf(fpLogFile, "\n");
      }

      if(j == 2)
      {
         fprintf(fpLogFile, "RECORD TYPE:\n");
         for(k = 0; k < pAnalysis->iNumTerms[j]; k++)
```

```
      {
         fprintf(fpLogFile, "%-20s%-12s%s\n",
               pAnalysis->term[i].szToken,
               numToken(pAnalysis->term[i].iTokenType),
               pAnalysis->term[i].szTerm);
         i++;
      }
      fprintf(fpLogFile, "\n");
   }

   if(j == 3)
   {
      fprintf(fpLogFile, "MEDIA TYPE:\n");
      for(k = 0; k < pAnalysis->iNumTerms[j]; k++)
      {
         fprintf(fpLogFile, "%-20s%-12s%s\n",
               pAnalysis->term[i].szToken,
               numToken(pAnalysis->term[i].iTokenType),
               pAnalysis->term[i].szTerm);
         i++;
      }
      fprintf(fpLogFile, "\n");
   }

   if(j == 4)
   {
      fprintf(fpLogFile, "RECORD FORMAT:\n");
      for(k = 0; k < pAnalysis->iNumTerms[j]; k++)
      {
         fprintf(fpLogFile, "%-20s%-12s%s\n",
               pAnalysis->term[i].szToken,
               numToken(pAnalysis->term[i].iTokenType),
               pAnalysis->term[i].szTerm);
         i++;
      }
      fprintf(fpLogFile, "\n");
   }
}

fprintf(fpLogFile, "\n*****************************");
fprintf(fpLogFile, " RECORD TEMPLATE ");
fprintf(fpLogFile, "*****************************\n");

for(i = 0; i < 5; i++)
{
   if(i == 0)
   {
      fprintf(fpLogFile, "SUBJECT:\n");
      for(j = 0; pCurRec->pSub[j].szKwrd[0] != '\0'; j++)
      {
         fprintf(fpLogFile, "Kwrd %-5d%-18sFreq = %d\n", j,
               pCurRec->pSub[j].szKwrd, pCurRec->pSub[j].iFreq);
      }
      fprintf(fpLogFile, "\n");
   }

   if(i == 1)
   {
      fprintf(fpLogFile, "ORIGINATING ORGANIZATION:\n");
      for(j = 0; pCurRec->pOrg[j].szKwrd[0] != '\0'; j++)
      {
         fprintf(fpLogFile, "Kwrd %-5d%-18sFreq = %d\n", j,
               pCurRec->pOrg[j].szKwrd, pCurRec->pOrg[j].iFreq);
      }
      fprintf(fpLogFile, "\n");
```

```
    }

    if(i == 2)
    {
       fprintf(fpLogFile, "RECORD TYPE:\n");
       for(j = 0; pCurRec->pTyp[j].szKwrd[0] != '\0'; j++)
       {
          fprintf(fpLogFile, "Kwrd %-5d%-18sFreq = %d\n", j,
               pCurRec->pTyp[j].szKwrd, pCurRec->pTyp[j].iFreq);
       }
       fprintf(fpLogFile, "\n");
    }

    if(i == 3)
    {
       fprintf(fpLogFile, "MEDIA TYPE:\n");
       for(j = 0; pCurRec->pMed[j].szKwrd[0] != '\0'; j++)
       {
          fprintf(fpLogFile, "Kwrd %-5d%-18sFreq = %d\n", j,
               pCurRec->pMed[j].szKwrd, pCurRec->pMed[j].iFreq);
       }
       fprintf(fpLogFile, "\n");
    }

    if(i == 4)
    {
       fprintf(fpLogFile, "RECORD FORMAT:\n");
       for(j = 0; pCurRec->pFrm[j].szKwrd[0] != '\0'; j++)
       {
          fprintf(fpLogFile, "Kwrd %-5d%-18sFreq = %d\n", j,
               pCurRec->pFrm[j].szKwrd, pCurRec->pFrm[j].iFreq);
       }
       fprintf(fpLogFile, "\n");
    }
}

fprintf(fpLogFile, "\n*****************************");
fprintf(fpLogFile, " SCORING RESULTS ");
fprintf(fpLogFile, "*****************************\n");
fprintf(fpLogFile, "30/20/30/10/10          ");
fprintf(fpLogFile, "20/20/20/20/20          ");
fprintf(fpLogFile, "50/30/00/10/10\n");
fprintf(fpLogFile, "DBF RANK SCORE          ");
fprintf(fpLogFile, "DBF RANK SCORE          ");
fprintf(fpLogFile, "DBF RANK SCORE\n");

(struct SCORE *)score = pScore;

for(i = 0; i < 5; i++)
{
   for(j = 0; j < 3; j++)
   {
      fprintf(fpLogFile, " %d   %d   %.3f          ", pScore->iDBFNum,
        pScore->iRank[j], pScore->fScore[j]);
   }
   fprintf(fpLogFile, "\n");
   pScore++;
}
fprintf(fpLogFile, "\n");

/* Reset pScore to the first element in the array */
pScore = score;

/* Record the details of the MOC caculations to logfile.txt */
for(i = 0; i < 5; i++)
```

```
{
   fprintf(fpLogFile, "%d SUB %2.0f / (%2d * %2.0f) = %5.3f",
      pScore->iDBFNum, pScore->sub.fTop, pScore->sub.iNumRecs,
      pScore->sub.fBottom, pScore->sub.fResult);
   fprintf(fpLogFile, " (0.3 = %5.3f) (0.2 = %5.3f) (0.5 = %5.3f)\n",
      pScore->sub.fResult * .3, pScore->sub.fResult * .2,
      pScore->sub.fResult * .5);

   fprintf(fpLogFile, "%d ORG %2.0f / (%2d * %2.0f) = %5.3f",
      pScore->iDBFNum, pScore->org.fTop, pScore->org.iNumRecs,
      pScore->org.fBottom, pScore->org.fResult);
   fprintf(fpLogFile, " (0.2 = %5.3f) (0.2 = %5.3f) (0.3 = %5.3f)\n",
      pScore->org.fResult * .2, pScore->org.fResult * .2,
      pScore->org.fResult * .3);

   fprintf(fpLogFile, "%d TYP %2.0f / (%2d * %2.0f) = %5.3f",
      pScore->iDBFNum, pScore->typ.fTop, pScore->typ.iNumRecs,
      pScore->typ.fBottom, pScore->typ.fResult);
   fprintf(fpLogFile, " (0.3 = %5.3f) (0.2 = %5.3f) (0.0 = %5.3f)\n",
      pScore->typ.fResult * .3, pScore->typ.fResult * .2,
      pScore->typ.fResult * .0);

   fprintf(fpLogFile, "%d MED %2.0f / (%2d * %2.0f) = %5.3f",
      pScore->iDBFNum, pScore->med.fTop, pScore->med.iNumRecs,
      pScore->med.fBottom, pScore->med.fResult);
   fprintf(fpLogFile, " (0.1 = %5.3f) (0.2 = %5.3f) (0.1 = %5.3f)\n",
      pScore->med.fResult * .1, pScore->med.fResult * .2,
      pScore->med.fResult * .1);

   fprintf(fpLogFile, "%d FRM %2.0f / (%2d * %2.0f) = %5.3f",
      pScore->iDBFNum, pScore->frm.fTop, pScore->frm.iNumRecs,
      pScore->frm.fBottom, pScore->frm.fResult);
   fprintf(fpLogFile, " (0.1 = %5.3f) (0.2 = %5.3f) (0.1 = %5.3f)\n",
      pScore->frm.fResult * .1, pScore->frm.fResult * .2,
      pScore->frm.fResult * .1);

   fprintf(fpLogFile, "                              Totals: ");
   for(j = 0; j < 3; j++)
   {
      fprintf(fpLogFile, "%5.3f         ", pScore->fScore[j]);
   }
   fprintf(fpLogFile, "\n\n");
   pScore++;
}

/* Reset pScore to the first element in the array */
pScore = score;

/* Move pScore pointer to the correct element */
for( ; pScore->iDBFNum != iDBFNum; pScore++);

for(i = 0; i < 3; i++)
{
   iRank[i] = pScore->iRank[i];
}

/* Determine number of duplicates of correct DBF */
for(i = 0; i < 3; i++)
{
   pScore = score;
   iDuplicate[i] = 0;
   for(j = 0; j < 5; j++)
   {
      if(pScore->iRank[i] == iRank[i] && pScore->iDBFNum != iDBFNum)
      {
```

```
            iDuplicate[i] += 1;
          }
          pScore++;
        }
    }

    /* Reset pScore to the first element in the array */
    pScore = score;

    /* Move pScore pointer to the correct element */
    for( ; pScore->iDBFNum != iDBFNum; pScore++);

    /* Log correct DBF and offsets to logfile.txt */
    fprintf(fpLogFile, "Correct DBF: %d  Offsets: %d  %d  %d\n\n",
       iDBFNum, (pScore->iRank[0] - 1 + iDuplicate[0]),
       (pScore->iRank[1] - 1 + iDuplicate[1]),
       (pScore->iRank[2] - 1 + iDuplicate[2]));

    /* Log correct DBF and offsets to scorelog.txt */
    fprintf(fpScoreLog, "DBF: %d  Offsets: %d   %d   %d\n",
       iDBFNum, (pScore->iRank[0] - 1 + iDuplicate[0]),
       (pScore->iRank[1] - 1 + iDuplicate[1]),
       (pScore->iRank[2] - 1 + iDuplicate[2]));

    fclose(fpLogFile);
    fclose(fpScoreLog);
}
```

```
/*-------------------------------------------------------------------------
PROJECT: racs.prj

FILE: db3funct.c

PURPOSE:
  This module contains functions to create and update dBASE III compatible
  files.

FUNCTIONS:
  void createDBF(int iDBFNum)
  void addRecord(int iDBFNum, struct DB3RECORD *pdb3record)
  void displayRecords(int iDBFNum)
  void editRecord(struct DB3RECORD *pdb3record, struct DB3HEADER *pdb3header,
     int iRecNum)
  void compactDBF(int iDBFNum)
-------------------------------------------------------------------------*/

#define EXTERN extern
#include "racs.h"


void createDBF(int iDBFNum)
{
   FILE *fpCurDBF;
   int i;
   char szHeader[20] = "dbfile";
   char szBuff[256];

   /* Create an instance of structure type struct DB3HEADER */
   struct DB3HEADER db3header;

   /* Create an array of nine sturtures of type COLUMNDEF */
   struct COLUMNDEF columnDef[9];

   /* Create an instance of structure type DB3RECORD */
   struct DB3RECORD db3record;

   struct tm *curTime;
   time_t tClock;

   /* Create new dBASE file */
   if((fpCurDBF = fopen(szpGetConfig(szHeader, iDBFNum), "wb")) == NULL)
   {
      displayError("could not create database file");
   }

   /* Get current time header information */
   time(&tClock);
   curTime = localtime(&tClock);

   /* Clear a block of memory for and initialize the db3header */
   memset(&db3header, 0, sizeof(db3header));
   db3header.bfVersion = 3;
   db3header.bfHasMemo = 0;
   db3header.bYear = curTime->tm_year;
   db3header.bMonth = (unsigned char)(curTime->tm_mon + 1);
   db3header.bDay  = (unsigned char)curTime->tm_mday;
   db3header.lNumberRecords = 0;
   db3header.nFirstRecordOffset = sizeof(db3header) + sizeof(columnDef) + 2;
   db3header.nRecordLength = sizeof(db3record);

   if((fwrite((char *)&db3header, sizeof(struct DB3HEADER),
      1, fpCurDBF)) != 1)
   {
```

```
      displayError("write error (database header)");
   }

   /* Zero-out memory and initialize the nine column definitions */
   memset(columnDef, 0, sizeof(columnDef));

   strcpy(columnDef[0].szColumnName, "DateRecord");
   columnDef[0].chType = CHARACTER_FIELD;
   columnDef[0].byLength = sizeof(db3record.szDateRecord);
   columnDef[0].byDecimalPlace = 0;

   strcpy(columnDef[1].szColumnName, "To");
   columnDef[1].chType = CHARACTER_FIELD;
   columnDef[1].byLength = sizeof(db3record.szTo);
   columnDef[1].byDecimalPlace = 0;

   strcpy(columnDef[2].szColumnName, "OriginOrg");
   columnDef[2].chType = CHARACTER_FIELD;
   columnDef[2].byLength = sizeof(db3record.szOriginOrg);
   columnDef[2].byDecimalPlace = 0;

   strcpy(columnDef[3].szColumnName, "Subject");
   columnDef[3].chType = CHARACTER_FIELD;
   columnDef[3].byLength = sizeof(db3record.szSubject);
   columnDef[3].byDecimalPlace = 0;

   strcpy(columnDef[4].szColumnName, "Author");
   columnDef[4].chType = CHARACTER_FIELD;
   columnDef[4].byLength = sizeof(db3record.szAuthor);
   columnDef[4].byDecimalPlace = 0;

   strcpy(columnDef[5].szColumnName, "CreateDate");
   columnDef[5].chType = CHARACTER_FIELD;
   columnDef[5].byLength = sizeof(db3record.szCreateDate);
   columnDef[5].byDecimalPlace = 0;

   strcpy(columnDef[6].szColumnName, "RecType");
   columnDef[6].chType = CHARACTER_FIELD;
   columnDef[6].byLength = sizeof(db3record.szRecType);
   columnDef[6].byDecimalPlace = 0;

   strcpy(columnDef[7].szColumnName, "MediaType");
   columnDef[7].chType = CHARACTER_FIELD;
   columnDef[7].byLength = sizeof(db3record.szMediaType);
   columnDef[7].byDecimalPlace = 0;

   strcpy(columnDef[8].szColumnName, "RecFormat");
   columnDef[8].chType = CHARACTER_FIELD;
   columnDef[8].byLength = sizeof(db3record.szRecFormat);
   columnDef[8].byDecimalPlace = 0;

   if((fwrite((char *)columnDef, sizeof(columnDef), 1, fpCurDBF)) != 1)
   {
      displayError("write error (column headers)");
   }

   if((fwrite((char *)"\r\0", sizeof(char) * 2, 1, fpCurDBF)) != 1)
   {
      displayError("write error (column headers)");
   }

   fclose(fpCurDBF);
}
```

```
void addRecord(int iDBFNum, struct DB3RECORD *pdb3record)
{
   FILE *fpCurDBF;

   int i;
   int iOffset;

   char szHeader[20] = "dbfile";

   struct tm *curTime;
   time_t tClock;

   struct DB3HEADER db3header;
   char *pcdb3record;

   clrscr();

   pcdb3record = (char *)pdb3record;

   /* Replace any NULLs with blank spaces for dBASE III compatibility */
   for (i = 0; i < sizeof(struct DB3RECORD); i++)
   {
      if (pcdb3record[i] == '\0')
      {
         pcdb3record[i] = ' ';
      }
   }

   /* Check to insure the intended database exists already */
   if((fpCurDBF = fopen(szpGetConfig(szHeader, iDBFNum), "rb")) == NULL)
   {
      displayError("database not initialized");
   }
   else
   {
      fclose(fpCurDBF);
   }

   if((fpCurDBF = fopen(szpGetConfig(szHeader, iDBFNum), "rb+")) == NULL)
   {
      displayError("could not open specified database");
   }

   if((fread(&db3header, sizeof(struct DB3HEADER), 1, fpCurDBF)) == NULL)
   {
      displayError("read error (database header)");
   }

   /* Set position for new db3record */
   iOffset = db3header.nFirstRecordOffset;
   iOffset += ((db3header.lNumberRecords) * (db3header.nRecordLength));
   fseek(fpCurDBF, iOffset, SEEK_SET);

   if((fwrite(pdb3record, sizeof(struct DB3RECORD), 1, fpCurDBF)) != 1)
   {
      displayError("write error (new db3record)");
   }

   /* Update values in database header */
   ++db3header.lNumberRecords;

   time(&tClock);
   curTime = localtime(&tClock);
   db3header.bYear = curTime->tm_year;
   db3header.bMonth = (unsigned char)(curTime->tm_mon + 1);
```

```
   db3header.bDay = (unsigned char)curTime->tm_mday;

   if((fseek(fpCurDBF, 0, SEEK_SET)) != 0)
   {
      displayError("seek error (rewirte of header)");
   }

   if((fwrite((char *)&db3header, sizeof(struct DB3HEADER), 1, fpCurDBF))
      != 1)
   {
      displayError("write error (updating header)");
   }
   fclose(fpCurDBF);
}


void displayRecords(int iDBFNum)
{
   FILE *fpCurDBF;
   int i, j;
   int iNext = FALSE;
   char cSel;
   char szStatus[8];
   char szHeader[20] = "dbfile";
   char *pBuff;

   /* Create an instance of structure type struct DB3HEADER */
   struct DB3HEADER db3header;
   struct DB3HEADER *pdb3header;

   /* Create an instance of structure type DB3RECORD */
   struct DB3RECORD db3record;
   struct DB3RECORD *pdb3record;

   pdb3header = &db3header;
   pdb3record = &db3record;

   if((fpCurDBF = fopen(szpGetConfig(szHeader, iDBFNum), "rb+")) == NULL)
   {
      displayError("error opening database for display/editing");
   }

   if((fread(&db3header, sizeof(struct DB3HEADER), 1, fpCurDBF)) == NULL)
   {
      displayError("read error (database header)");
   }

   fseek(fpCurDBF, db3header.nFirstRecordOffset, SEEK_SET);

   i = 1;
   while( i <= db3header.lNumberRecords)
   {
      clrscr();

      fseek(fpCurDBF, (db3header.nFirstRecordOffset +
                ((i - 1) * db3header.nRecordLength)), SEEK_SET);
      if((fread(&db3record, sizeof(struct DB3RECORD), 1, fpCurDBF)) == NULL)
      {
         displayError("read error (database record)");
      }

      for(j = 25; db3record.szDateRecord[j] == ' ' && j != 0; j--);
      j++;
      db3record.szDateRecord[j] = '\0';
```

```
for(j = 100; db3record.szTo[j] == ' ' && j != 0; j--);
j++;
db3record.szTo[j] = '\0';

for(j = 100; db3record.szOriginOrg[j] == ' ' && j != 0; j--);
j++;
db3record.szOriginOrg[j] = '\0';

for(j = 254; db3record.szSubject[j] == ' ' && j != 0; j--);
j++;
db3record.szSubject[j] = '\0';

for(j = 100; db3record.szAuthor[j] == ' ' && j != 0; j--);
j++;
db3record.szAuthor[j] = '\0';

for(j = 25; db3record.szCreateDate[j] == ' ' && j != 0; j--);
j++;
db3record.szCreateDate[j] = '\0';

for(j = 50; db3record.szRecType[j] == ' ' && j != 0; j--);
j++;
db3record.szRecType[j] = '\0';

for(j = 50; db3record.szMediaType[j] == ' ' && j != 0; j--);
j++;
db3record.szMediaType[j] = '\0';

for(j = 50; db3record.szRecFormat[j] == ' ' && j != 0; j--);
j++;
db3record.szRecFormat[j] = '\0';

iNext = FALSE;
while(iNext == FALSE)
{
   clrscr();
   if(db3record.szStatus[0] == USABLE_RECORD)
   {
      strcpy(szStatus, "active");
   }
   else if(db3record.szStatus[0] == DELETED_RECORD)
   {
      strcpy(szStatus, "deleted");
   }
   else
   {
      strcpy(szStatus, "unknown");
   }
   _setcursortype(_NOCURSOR);
   printf("Record %d of %d\n", i, db3header.lNumberRecords);
   printf("Date of Record: %s\t", pdb3record->szDateRecord);
   printf("Status: %s\n", szStatus);
   printf("- 1 - Addressee(s):  %-100s\n", pdb3record->szTo);
   printf("- 2 - ORIGINATOR:  %-100s\n", pdb3record->szOriginOrg);
   printf("- 3 - SUBJECT:  %-254s\n", pdb3record->szSubject);
   printf("- 4 - Author:  %-100s\n", pdb3record->szAuthor);
   printf("- 5 - Creation Date:  %-25s\n\n",
        pdb3record->szCreateDate);
   printf("- 6 - RECORD TYPE:  %-50s\n\n", pdb3record->szRecType);
   printf("- 7 - MEDIA TYPE:  %-50s\n\n", pdb3record->szMediaType);
   printf("- 8 - RECORD FORMAT:  %-50s\n\n",
        pdb3record->szRecFormat);
   printf("\n(e) %-10s(n) %-10s(p) %-10s(f) %-10s(l) %-10s\n",
         "Edit", "Next", "Prev", "First", "Last");
   puts("(q) Return to previous menu");
```

```
cSel = getch();

switch(cSel)
{
   case 'e':
   case 'E':
      editRecord(pdb3record, pdb3header, i);
      fseek(fpCurDBF, (db3header.nFirstRecordOffset +
          ((i - 1) * db3header.nRecordLength)), SEEK_SET);
      if((fwrite(pdb3record, sizeof(struct DB3RECORD),
         1, fpCurDBF)) != 1)
      {
         displayError("write error (edited db3record)");
      }
      fseek(fpCurDBF, 0, SEEK_SET);
      if((fwrite(pdb3header, sizeof(struct DB3HEADER),
         1, fpCurDBF)) != 1)
      {
         displayError("write error (DB3 Header)");
      }
      iNext = TRUE;
      break;

   case 'n':
   case 'N':
      iNext = TRUE;
      if(i != db3header.lNumberRecords)
      {
         i++;
      }
      else
      {
         puts("AT LAST RECORD!");
         delay(500);
      }
      break;

   case 'p':
   case 'P':
      iNext = TRUE;
      if(i != 1)
      {
         i--;
      }
      else
      {
         puts("AT FIRST RECORD!");
         delay(500);
      }
      break;

   case 'f':
   case 'F':
      iNext = TRUE;
      i = 1;
      break;

   case 'l':
   case 'L':
      i = db3header.lNumberRecords;
      iNext = TRUE;
      break;

   case 'q':
   case 'Q':
```

```
            return;

         default:
            puts("INVALID KEY!");
            delay(500);
      }
   }
}
fclose(fpCurDBF);
}


void editRecord(struct DB3RECORD *pdb3record, struct DB3HEADER *pdb3header,
          int iRecNum)
{
   int i;
   int iDone = FALSE;

   char cSel;
   char szBuff[255];
   char szRecStatus[8];
   char *pcdb3record;

   struct tm *curTime;
   time_t tClock;

   pcdb3record = (char *)pdb3record;

   clrscr();

   time(&tClock);
   curTime = localtime(&tClock);
   strftime(szBuff, 255, "%d-%b-%Y %X", curTime);
   strncpy(pdb3record->szDateRecord, szBuff,
        sizeof(pdb3record->szDateRecord));

   cSel = 0;
   while(iDone == FALSE)
   {
      _setcursortype(_NORMALCURSOR);
      if(pdb3record->szStatus[0] == DELETED_RECORD)
      {
         strcpy(szRecStatus, "deleted");
      }
      else
      {
         strcpy(szRecStatus, "active");
      }

      switch(cSel)
      {
         case 49:
            printf("Addressee(s): ");
            gets(szBuff);
            strncpy(pdb3record->szTo, szBuff,
                 sizeof(pdb3record->szTo));
            cSel = 0;
            break;

         case 50:
            printf("ORIGINATING ORGANIZATION: ");
            gets(szBuff);
            strncpy(pdb3record->szOriginOrg, szBuff,
                 sizeof(pdb3record->szOriginOrg));
            cSel = 0;
```

```
      break;

   case 51:
      printf("SUBJECT: ");
      gets(szBuff);
      strncpy(pdb3record->szSubject, szBuff,
            sizeof(pdb3record->szSubject));
      cSel = 0;
      break;

   case 52:
      printf("Author: ");
      gets(szBuff);
      strncpy(pdb3record->szAuthor, szBuff,
            sizeof(pdb3record->szAuthor));
      cSel = 0;
      break;

   case 53:
      printf("Creation Date: ");
      gets(szBuff);
      strncpy(pdb3record->szCreateDate, szBuff,
            sizeof(pdb3record->szCreateDate));
         cSel = 0;
      break;

   case 54:
      printf("RECORD TYPE: ");
      gets(szBuff);
      strncpy(pdb3record->szRecType, szBuff,
            sizeof(pdb3record->szRecType));
      cSel = 0;
      break;

   case 55:
      printf("MEDIA TYPE: ");
      gets(szBuff);
      strncpy(pdb3record->szMediaType, szBuff,
            sizeof(pdb3record->szMediaType));
      cSel = 0;
      break;

   case 56:
      printf("RECORD FORMAT: ");
      gets(szBuff);
      strncpy(pdb3record->szRecFormat, szBuff,
            sizeof(pdb3record->szRecFormat));
      cSel = 0;
      break;

   case 's':
   case 'S':
      iDone = TRUE;
      break;

   case 'd':
   case 'D':
      pdb3record->szStatus[0] = DELETED_RECORD;
      cSel = 0;
      break;

   case 'u':
   case 'U':
      pdb3record->szStatus[0] = USABLE_RECORD;
      cSel = 0;
```

```
            break;

         default:
            clrscr();
            _setcursortype(_NOCURSOR);
            printf("Record %d of %d\n", iRecNum,
                  pdb3header->lNumberRecords);
            printf("Date of Record: %s\t",
                  pdb3record->szDateRecord);
            printf("Status: %s\n", szRecStatus);
            printf("- 1 - Addressee(s):  %-100s\n",
                  pdb3record->szTo);
            printf("- 2 - ORIGINATOR:  %-100s\n",
                  pdb3record->szOriginOrg);
            printf("- 3 - SUBJECT:  %-254s\n",
                  pdb3record->szSubject);
            printf("- 4 - Author:  %-100s\n",
                  pdb3record->szAuthor);
            printf("- 5 - Creation Date:  %-25s\n\n",
                  pdb3record->szCreateDate);
            printf("- 6 - RECORD TYPE:  %-50s\n\n",
                  pdb3record->szRecType);
            printf("- 7 - MEDIA TYPE:  %-50s\n\n",
                  pdb3record->szMediaType);
            printf("- 8 - RECORD FORMAT:  %-50s\n\n",
                  pdb3record->szRecFormat);
            puts("\nTo reenter any fields enter the appropriate number");
            printf("(s) %-10s(d) %-10s(u) %-10s\n",
                  "Save", "Del", "Undelete");
            cSel = getch();
      }
   }
   _setcursortype(_NOCURSOR);

   for (i = 0; i < sizeof(struct DB3RECORD); i++)
   {
      if (pcdb3record[i] == '\0')
      {
         pcdb3record[i] = ' ';
      }
   }

   time(&tClock);
   curTime = localtime(&tClock);
   pdb3header->bYear = curTime->tm_year;
   pdb3header->bMonth = (unsigned char)(curTime->tm_mon + 1);
   pdb3header->bDay = (unsigned char)curTime->tm_mday;
}


void compactDBF(int iDBFNum)
{
   FILE *fpCurDBF;
   FILE *fpTmpDBF;

   int i;
   char c;
   char szHeader[20] = "dbfile";
   char szBuff[256];

   struct DB3HEADER db3headOld;
   struct DB3HEADER db3headNew;

   struct DB3RECORD db3record;
```

```
struct tm *curTime;
time_t tClock;

createDBF(6);

if((fpCurDBF = fopen(szpGetConfig(szHeader, iDBFNum), "rb")) == NULL)
{
   displayError("opening DBF file for compacting");
}

if((fpTmpDBF = fopen(szpGetConfig(szHeader, 6), "rb+")) == NULL)
{
   displayError("opening temp file for compacting");
}

if((fread(&db3headOld, sizeof(struct DB3HEADER), 1, fpCurDBF)) == NULL)
{
   displayError("read error (database header)");
}

if((fread(&db3headNew, sizeof(struct DB3HEADER), 1, fpTmpDBF)) == NULL)
{
   displayError("read error (database header)");
}

i = 1;
while(i <= db3headOld.lNumberRecords)
{
   fseek(fpCurDBF, (db3headOld.nFirstRecordOffset +
      ((i - 1) * db3headOld.nRecordLength)), SEEK_SET);
   if((fread(&db3record, sizeof(struct DB3RECORD), 1, fpCurDBF)) == NULL)
   {
      displayError("read error (database record)");
   }
   if(db3record.szStatus[0] != '*')
   {
      fseek(fpTmpDBF, (db3headNew.nFirstRecordOffset +
         (db3headNew.lNumberRecords * db3headNew.nRecordLength)),
         SEEK_SET);
      if((fwrite((char *)&db3record, sizeof(struct DB3RECORD), 1, fpTmpDBF))
         != 1)
      {
         displayError(":( write error (DB3 record in temp file)");
      }
      db3headNew.lNumberRecords++;
   }
   i++;
}
time(&tClock);
curTime = localtime(&tClock);
db3headNew.bYear = curTime->tm_year;
db3headNew.bMonth = (unsigned char)(curTime->tm_mon + 1);
db3headNew.bDay = (unsigned char)curTime->tm_mday;

if((fseek(fpTmpDBF, 0, SEEK_SET)) != 0)
{
   displayError("seek error (rewirte of header in temp file)");
}

if((fwrite(&db3headNew, sizeof(struct DB3HEADER), 1, fpTmpDBF)) != 1)
{
   displayError("write error (updating header in temp file)");
}
fclose(fpTmpDBF);
fclose(fpCurDBF);
```

```
   if((fpTmpDBF = fopen(szpGetConfig(szHeader, 6), "rb")) == NULL)
   {
      displayError("opening temp file for copying");
   }

   if((fpCurDBF = fopen(szpGetConfig(szHeader, iDBFNum), "wb")) == NULL)
   {
      displayError("opening current DBF file for compacting");
   }

   while(1)
   {
      c = fgetc(fpTmpDBF);

      if(!feof(fpTmpDBF))
         fputc(c, fpCurDBF);
      else
         break;
   }

   fclose(fpTmpDBF);
   fclose(fpCurDBF);
   remove(szpGetConfig(szHeader, 6));
   printf("\t\t  Database %d compacted\n", iDBFNum);
}
```

```
/*-------------------------------------------------------------------------
PROJECT: racs.prj

FILE: main.c

PURPOSE:
  Start and end point for program as well as general utility functions.

FUNCTIONS:
  void main(void)
  char *szpGetConfig(char szHeaderText[], int iFileNum)
  void displayError(char szErrorMessage[])
  void copyFile(char *oldName, char *newName)
  void cleanUp(void)
-------------------------------------------------------------------------*/

#define EXTERN extern
#include "racs.h"


void main(void)
{
   _setcursortype(_NOCURSOR);
   atexit(cleanUp);

   introScreen();
   mainMenu();

   exit(0);
}


char *szpGetConfig(char szHeaderText[], int iFileNum)
{
   char szHeader[20] = "";
   char szLBracket[] = "[";
   char szRBracket[] = "]";
   char szBuff[81] = "";
   char szFileName[81] = "";
   char szErrorMessage[81] = "";

   int i;

   FILE *fpConfig;

   strcat(szHeader, szLBracket);
   strcat(szHeader, strupr(szHeaderText));
   strcat(szHeader, szRBracket);

   if ((fpConfig = fopen("CONFIG.TXT", "r")) == NULL)
   {
      displayError("config.txt not found");
   }

   while(strcmp(szBuff, szHeader) != 0)
   {
      fscanf(fpConfig, "%s", szBuff);
      if(strcmp(szBuff, "[END]") == 0)
      {
         strcpy(szErrorMessage, szHeader);
         strcat(szErrorMessage, " not found in config.txt");
         displayError(szErrorMessage);
      }
   }
```

```
   for(i = 0; i < iFileNum; i++)
   {
      fscanf(fpConfig, "%s", szFileName);
      if((strcmp(szBuff, "[END]") == 0) || (szFileName[0] == '['))
      {
         strcpy(szErrorMessage, "specified file not found under ");
         strcat(szErrorMessage, szHeader);
         displayError(szErrorMessage);
      }
   }
   fclose(fpConfig);
   return szFileName;
}


void displayError(char szErrorMessage[])
{
   clrscr();
   puts("\n\n\n\n\n");
   printf("\t\t ERROR: %s", szErrorMessage);
   delay(3000);
   exit(1);
}

void copyFile(char *oldName, char *newName)
{
   FILE *fpOld, *fpNew;
   int c;

   if((fpOld = fopen(oldName, "rb")) == NULL)
   {
      displayError("opening file to backup");
   }

   if((fpNew = fopen(newName, "wb")) == NULL)
   {
      displayError("backup could not be created");
   }

   while(1)
   {
      c = fgetc(fpOld);

      if(!feof(fpOld))
         fputc(c, fpNew);
      else
         break;
   }
   fclose(fpOld);
   fclose(fpNew);
}


void cleanUp(void)
{
   clrscr();
   puts("\n\n\n\n\n");
   puts("\t\t\t\t Goodbye!");
   delay(1000);
   fcloseall();
   clrscr();
   _setcursortype(_NORMALCURSOR);
}
```

```
/*-------------------------------------------------------------------------
PROJECT: racs.prj

FILE: menus.c

PURPOSE:
  Contains all the functions which display the various menus
  needed to operate the program.

FUNCTIONS:
  void introScreen(void)
  void mainMenu(void)
  void databaseMenu(void)
  void initializeDatabaseMenu(void)
  void viewDatabaseMenu(void)
  void compactDatabaseMenu(void)
  void initializeDBF(int iDBFNum)
  void templateMenu(void)
  void generateTemplatesMenu(void)
  void viewTemplatesMenu(void)
  void logFileMenu(void)
  void viewLogFileMenu(void)
-------------------------------------------------------------------------*/

#define EXTERN extern
#include "racs.h"

void introScreen(void)
{
   clrscr();
   window(16, 6, 65, 13);
   textbackground(BLUE);
   textcolor(LIGHTGRAY);
   clrscr();
   cprintf("\r\n");
   cprintf("                    R.A.C.S.                        \r\n");
   cprintf("    Records Analysis and Classification System      \r\n");
   cprintf("\r\n");
   cprintf("                  Version 1.0                       \r\n");
   cprintf("            Created by David Snoddy                 \r\n");
   cprintf("                  October 1996                      \r\n");
   delay(3500);
   window(1, 1, 80, 25);
   textbackground(BLACK);
   textcolor(LIGHTGRAY);
   clrscr();
}

void mainMenu(void)
{
   int iDone = FALSE;
   char cSel;

   while(iDone == FALSE)
   {
      clrscr();
      puts("\n\n");
      puts("\t\t\tChoose one of the following actions:\n");
      puts("\t\t\t  (d)\tDatabase Management");
      puts("\t\t\t  (t)\tTemplate Management");
      puts("\t\t\t  (l)\tLog File Management");
      puts("\t\t\t  (c)\tClassify New Record");
      puts("");
      puts("\t\t\t  (q)\tQuit");
```

```
      cSel = getch();

      switch(cSel)
      {
         case 'd':
         case 'D':
            databaseMenu();
            break;

         case 't':
         case 'T':
            templateMenu();
            break;

         case 'l':
         case 'L':
            logFileMenu();
            break;

         case 'c':
         case 'C':
            classifyRecord();
            break;

         case 'q':
         case 'Q':
            iDone = TRUE;
            break;

         default:
            puts("\n\t\t\t  INVALID KEY!");
            delay(1000);
      }
   }
}


void databaseMenu(void)
{
   int i;
   int iSuccess[5];
   int iDone = FALSE;
   char cSel;
   char szHeader1[20] = "dbfile";
   char szHeader2[20] = "dbfbackup";
   char szNew[20];
   char szOld[20];

   while(iDone == FALSE)
   {
      clrscr();
      puts("\n\n");
      puts("\t\t\tChoose one of the following actions:\n");
      puts("\t\t\t  (b)\tBackup All Databases");
      puts("\t\t\t  (i)\tInitialize Databases");
      puts("\t\t\t  (v)\tView/Edit Records");
      puts("\t\t\t  (c)\tCompact Databases");
      puts("");
      puts("\t\t\t  (q)\tReturn to the Main Menu");

      cSel = getch();

      switch(cSel)
      {
         case 'b':
```

```
         case 'B':
            for(i = 1; i <= 5; i++)
            {
               strcpy(szOld, szpGetConfig(szHeader1, i));
               strcpy(szNew, szpGetConfig(szHeader2, i));
               copyFile(szOld, szNew);
            }
            printf("\n\t\t\tAll databases backed up");
            delay(2000);
            break;

         case 'i':
         case 'I':
            initializeDatabaseMenu();
            break;

         case 'v':
         case 'V':
            viewDatabaseMenu();
            break;

         case 'c':
         case 'C':
            compactDatabaseMenu();
            break;

         case 'q':
         case 'Q':
            iDone = TRUE;
            break;

         default:
            puts("\n\t\t\t  INVALID KEY!");
            delay(1000);
      }
   }
}


void initializeDatabaseMenu(void)
{
   int iDone = FALSE;
   char cSel;

   while(iDone == FALSE)
   {
      _setcursortype(_NOCURSOR);
      clrscr();
      puts("\n\n");
      puts("\t\t\tSelect the database to initialize:\n");
      puts("\t\t\t  (1)\tT  11-02 R 21 Item 3");
      puts("\t\t\t  (2)\tT  11-01 R 01 Item 6-3-2");
      puts("\t\t\t  (3)\tT  11-01 R 01 Item 6-4");
      puts("\t\t\t  (4)\tT  11-02 R 33 Item 12");
      puts("\t\t\t  (5)\tT 900-02 R 02 Item 15");
      puts("\t\t\t  (a)\tInitialize All Databases");
      puts("");
      puts("\t\t\t  (q)\tReturn to the Databases Menu");

      cSel = getch();

      switch(cSel)
      {
         case '1':
            initializeDBF(1);
```

```
            break;

         case '2':
            initializeDBF(2);
            break;

         case '3':
            initializeDBF(3);
            break;

         case '4':
            initializeDBF(4);
            break;

         case '5':
            initializeDBF(5);
            break;

         case 'a':
         case 'A':
            initializeDBF('a');
            break;

         case 'q':
         case 'Q':
            iDone = TRUE;
            break;

         default:
            puts("\n\t\t\t  INVALID KEY!");
            delay(1000);
      }
   }
}


void viewDatabaseMenu(void)
{
   int iDone = FALSE;
   char cSel;

   while(iDone == FALSE)
   {
      _setcursortype(_NOCURSOR);
      clrscr();
      puts("\n\n");
      puts("\t\t\tSelect the database to view/edit:\n");
      puts("\t\t\t  (1)\tT  11-02 R 21 Item 3");
      puts("\t\t\t  (2)\tT  11-01 R 01 Item 6-3-2");
      puts("\t\t\t  (3)\tT  11-01 R 01 Item 6-4");
      puts("\t\t\t  (4)\tT  11-02 R 33 Item 12");
      puts("\t\t\t  (5)\tT 900-02 R 02 Item 15");
      puts("");
      puts("\t\t\t  (q)\tReturn to Databases Menu");

      cSel = getch();

      switch(cSel)
      {
         case '1':
            displayRecords(1);
            break;

         case '2':
            displayRecords(2);
```

```
            break;

         case '3':
            displayRecords(3);
            break;

         case '4':
            displayRecords(4);
            break;

         case '5':
            displayRecords(5);
            break;

         case 'q':
         case 'Q':
            iDone = TRUE;
            break;

         default:
            puts("\n\t\t\t  INVALID KEY!");
            delay(1000);
      }
   }
}


void compactDatabaseMenu(void)
{
   int iDone = FALSE;
   char cSel;

   while(iDone == FALSE)
   {
      _setcursortype(_NOCURSOR);
      clrscr();
      puts("\n\n");
      puts("\t\t\tSelect the database to compact:\n");
      puts("\t\t\t  (1)\tT  11-02 R 21 Item 3");
      puts("\t\t\t  (2)\tT  11-01 R 01 Item 6-3-2");
      puts("\t\t\t  (3)\tT  11-01 R 01 Item 6-4");
      puts("\t\t\t  (4)\tT  11-02 R 33 Item 12");
      puts("\t\t\t  (5)\tT 900-02 R 02 Item 15");
      puts("\t\t\t  (a)\tCompact All Databases");
      puts("");
      puts("\t\t\t  (q)\tReturn to the Databases Menu");

      cSel = getch();

      switch(cSel)
      {
         case '1':
            compactDBF(1);
            delay(500);
            break;

         case '2':
            compactDBF(2);
            delay(500);
            break;

         case '3':
            compactDBF(3);
            delay(500);
            break;
```

```
         case '4':
            compactDBF(4);
            delay(500);
            break;

         case '5':
            compactDBF(5);
            delay(500);
            break;

         case 'a':
         case 'A':
            compactDBF(1);
            compactDBF(2);
            compactDBF(3);
            compactDBF(4);
            compactDBF(5);
            delay(500);
            break;

         case 'q':
         case 'Q':
            iDone = TRUE;
            break;

         default:
            puts("\n\t\t\t  INVALID KEY!");
            delay(1000);
      }
   }
}


void initializeDBF(int iDBFNum)
{
   int i;
   int iSuccess;
   char cSel;
   char szHeader1[20] = "dbfile";
   char szHeader2[20] = "dbfbackup";
   char szOld[20];
   char szNew[20];

   puts("\n");
   puts("\t\t\tWARNING!");
   puts("\t\t\tInitializing a database will delete any previous");
   puts("\t\t\tinformation stored in the database!");
   puts("");
   puts("\t\t\tDo you wish to continue?");
   puts("\t\t\t(y) to continue any other key to abandon operation\n");

   cSel = getch();

   if(cSel == 'y' || cSel == 'Y')
   {
      if(iDBFNum == 'a')
      {
         for(i = 1; i <= 5; i++)
         {
            strcpy(szOld, szpGetConfig(szHeader1, i));
            strcpy(szNew, szpGetConfig(szHeader2, i));
            copyFile(szOld, szNew);
            createDBF(i);
         }
```

```
      }
      else
      {
         strcpy(szOld, szpGetConfig(szHeader1, iDBFNum));
         strcpy(szNew, szpGetConfig(szHeader2, iDBFNum));
         copyFile(szOld, szNew);
         createDBF(iDBFNum);
      }

      if(iDBFNum == 'a')
      {
         puts("\t\t\tAll databases initialized");
      }
      else
      {
         printf("\t\t\tDatabase %d initialized\n", iDBFNum);
      }
      delay(1000);
   }
   else
   {
      puts("\n\t\t\tOperation aborted!");
      delay(1000);
   }
}


void templateMenu(void)
{
   int i;
   int iDone = FALSE;
   char cSel;

   while(iDone == FALSE)
   {
      clrscr();
      puts("\n\n");
      puts("\t\t\tChoose one of the following actions:\n");
      puts("\t\t\t  (g)\tGenerate Templates");
      puts("\t\t\t  (v)\tView Templates");
      puts("");
      puts("\t\t\t  (q)\tReturn to the Main Menu");

      cSel = getch();

      switch(cSel)
      {
         case 'g':
         case 'G':
            generateTemplatesMenu();
            break;

         case 'v':
         case 'V':
            viewTemplatesMenu();
            break;

         case 'q':
         case 'Q':
            iDone = TRUE;
            break;

         default:
            puts("\n\t\t\t  INVALID KEY!");
            delay(1000);
```

```
      }
   }
}


void generateTemplatesMenu(void)
{
   int iDone = FALSE;
   int i;
   char cSel;

   while(iDone == FALSE)
   {
      _setcursortype(_NOCURSOR);
      clrscr();
      puts("\n\n");
      puts("\t\t\tSelect the template to generate:\n");
      puts("\t\t\t  (1)\tT  11-02 R 21 Item 3");
      puts("\t\t\t  (2)\tT  11-01 R 01 Item 6-3-2");
      puts("\t\t\t  (3)\tT  11-01 R 01 Item 6-4");
      puts("\t\t\t  (4)\tT  11-02 R 33 Item 12");
      puts("\t\t\t  (5)\tT 900-02 R 02 Item 15");
      puts("\t\t\t  (a)\tGenerate All Templates");
      puts("");
      puts("\t\t\t  (q)\tReturn to the Templates Menu");

      cSel = getch();

      switch(cSel)
      {
         case '1':
            if(genCLASSTMPLT(1) == TRUE);
            {
               printf("\n\t\t\tTemplate %c generated", cSel);
               delay(500);
            }
            break;

         case '2':
            if(genCLASSTMPLT(2) == TRUE);
            {
               printf("\n\t\t\tTemplate %c generated", cSel);
               delay(500);
            }
            break;

         case '3':
            if(genCLASSTMPLT(3) == TRUE);
            {
               printf("\n\t\t\tTemplate %c generated", cSel);
               delay(500);
            }
            break;

         case '4':
            if(genCLASSTMPLT(4) == TRUE);
            {
               printf("\n\t\t\tTemplate %c generated", cSel);
               delay(500);
            }
            break;

         case '5':
            if(genCLASSTMPLT(5) == TRUE);
            {
```

```
            printf("\n\t\t\tTemplate %c generated", cSel);
            delay(500);
         }
         break;

      case 'a':
      case 'A':
         puts("");
         for(i = 1; i <= 5; i++)
         {
            if(genCLASSTMPLT(i) == TRUE);
            {
               printf("\t\t\tTemplate %d generated\n", i);
               delay(100);
            }
         }
         break;

      case 'q':
      case 'Q':
         iDone = TRUE;
         break;

      default:
         puts("\n\t\t\t  INVALID KEY!");
         delay(1000);
   }
  }
}

void viewTemplatesMenu(void)
{
   int iDone = FALSE;
   int i;
   char cSel;

   while(iDone == FALSE)
   {
      _setcursortype(_NOCURSOR);
      clrscr();
      puts("\n\n");
      puts("\t\t\tSelect the template to view:\n");
      puts("\t\t\t  (1)\tT  11-02 R 21 Item 3");
      puts("\t\t\t  (2)\tT  11-01 R 01 Item 6-3-2");
      puts("\t\t\t  (3)\tT  11-01 R 01 Item 6-4");
      puts("\t\t\t  (4)\tT  11-02 R 33 Item 12");
      puts("\t\t\t  (5)\tT 900-02 R 02 Item 15");
      puts("");
      puts("\t\t\t  (q)\tReturn to the Templates Menu");

      cSel = getch();

      switch(cSel)
      {
         case '1':
            system("edit cat1tpl.txt");
            break;

         case '2':
            system("edit cat2tpl.txt");
            break;

         case '3':
            system("edit cat3tpl.txt");
```

```
            break;

         case '4':
            system("edit cat4tpl.txt");
            break;

         case '5':
            system("edit cat5tpl.txt");
            break;

         case 'q':
         case 'Q':
            iDone = TRUE;
            break;

         default:
            puts("\n\t\t\t  INVALID KEY!");
            delay(1000);
      }
   }
}


void logFileMenu(void)
{
   int i;
   int iDone = FALSE;
   char cSel;
   char szHeader1[20] = "logfile";
   char szHeader2[20] = "logbackup";
   char szOld[20];
   char szNew[20];

   while(iDone == FALSE)
   {
      clrscr();
      puts("\n\n");
      puts("\t\t\tChoose one of the following actions:\n");
      puts("\t\t\t  (b)\tBackup Log Files");
      puts("\t\t\t  (d)\tDelete Log Files");
      puts("\t\t\t  (v)\tView Log Files");
      puts("");
      puts("\t\t\t  (q)\tReturn to the Main Menu");

      cSel = getch();

      switch(cSel)
      {
         case 'b':
         case 'B':
            strcpy(szOld, szpGetConfig(szHeader1, 1));
            strcpy(szNew, szpGetConfig(szHeader2, 1));
            copyFile(szOld, szNew);
            strcpy(szOld, szpGetConfig(szHeader1, 2));
            strcpy(szNew, szpGetConfig(szHeader2, 2));
            copyFile(szOld, szNew);
            printf("\n\t\t\tLog files backed up");
            delay(750);
            break;

         case 'd':
         case 'D':
            strcpy(szOld, szpGetConfig(szHeader1, 1));
            strcpy(szNew, szpGetConfig(szHeader2, 1));
            copyFile(szOld, szNew);
```

```
            remove(szOld);
            strcpy(szOld, szpGetConfig(szHeader1, 2));
            strcpy(szNew, szpGetConfig(szHeader2, 2));
            copyFile(szOld, szNew);
            remove(szOld);
            printf("\n\t\t\tLog files deleted");
            delay(750);
            break;

         case 'v':
         case 'V':
            viewLogFileMenu();
            break;

         case 'q':
         case 'Q':
            iDone = TRUE;
            break;

         default:
            puts("\n\t\t\t  INVALID KEY!");
            delay(1000);
      }
   }
}


void viewLogFileMenu(void)
{
   int iDone = FALSE;
   int i;
   char cSel;

   while(iDone == FALSE)
   {
      _setcursortype(_NOCURSOR);
      clrscr();
      puts("\n\n");
      puts("\t\t\tSelect the log file to view:\n");
      puts("\t\t\t  (a)\tAll Details");
      puts("\t\t\t  (s)\tOnly Score");
      puts("");
      puts("\t\t\t  (q)\tReturn to the Log Files Menu");

      cSel = getch();

      switch(cSel)
      {
         case 'a':
         case 'A':
            system("edit logfile.txt");
            break;

         case 's':
         case 'S':
            system("edit scorelog.txt");
            break;

         case 'q':
         case 'Q':
            iDone = TRUE;
            break;

         default:
            puts("\n\t\t\t  INVALID KEY!");
```

```
                delay(1000);
            }
        }
    }
```

```
/*--------------------------------------------------------------------------
PROJECT: racs.prj

FILE: score.c

PURPOSE:
  The functions in this module perform the mathematical calculations to
  determine the scores for each new record.  The scores are determined by
  calculating a Modified Overlap Coefficient for the record template versus
  each of the five class templates.

FUNCTIONS:
  void compareTemplates(struct RECTMPLT *pRecTmplt, struct SCORE *pScore)
  void calcScore(struct RECTMPLT *pRecTmplt, struct CLASSTMPLT *pClsTmplt,
     struct SCORE *pScore)
--------------------------------------------------------------------------*/

#define EXTERN extern
#include "racs.h"


void compareTemplates(struct RECTMPLT *pRecTmplt, struct SCORE *pScore)
{
   FILE *fpClsTmplt;
   char szHeader[20] = "template";

   int i;

   struct CLASSTMPLT clsTmplt;
   struct CLASSTMPLT *pClsTmplt;
   pClsTmplt = &clsTmplt;

   for(i = 0; i < 5; i++)
   {
      if((fpClsTmplt = fopen(szpGetConfig(szHeader, i + 1), "rb")) == NULL)
      {
         displayError("opening class template");
      }

      memset(&clsTmplt, 0, sizeof(struct CLASSTMPLT));

      if((fread(&clsTmplt, sizeof(struct CLASSTMPLT), 1,
         fpClsTmplt)) == NULL)
      {
         displayError("read error (class template)");
      }

      fclose(fpClsTmplt);

      pScore->iDBFNum = (i + 1);
      calcScore(pRecTmplt, pClsTmplt, pScore);
      pScore++;
   }
}


void calcScore(struct RECTMPLT *pRecTmplt, struct CLASSTMPLT *pClsTmplt,
   struct SCORE *pScore)
{
   int i, j;
   int iMatch;
   float fTop;
   float fBottom;
   float fSub = 0;
   float fOrg = 0;
```

```
float fTyp = 0;
float fMed = 0;
float fFrm = 0;
float fScore[3] = {0, 0, 0};

/* Caculate score for the subject fields */
iMatch = FALSE;
fTop = 0;
fBottom = 0;
for(i = 0; (pRecTmplt->pSub[i].szKwrd[0] != '\0'); i++)
{
   iMatch = FALSE;
   for(j = 0; (pClsTmplt->pSub[j].szKwrd[0] != '\0' &&
      iMatch != TRUE); j++)
   {
      clrscr();
      printf("Template %d - Subject\n", pScore->iDBFNum);
      printf("  Record: %s \n", pRecTmplt->pSub[i].szKwrd);
      printf("Template: %s \n", pClsTmplt->pSub[j].szKwrd);
      delay(5);

      if(strcmp(pRecTmplt->pSub[i].szKwrd,
            pClsTmplt->pSub[j].szKwrd) == 0)
      {
         iMatch = TRUE;
         fTop += ((pRecTmplt->pSub[i].iFreq) *
            (pClsTmplt->pSub[j].iFreq));
         fBottom += pRecTmplt->pSub[i].iFreq;
         printf("record freq: %d\n", pRecTmplt->pSub[i].iFreq);
         printf(" class freq: %d\n", pClsTmplt->pSub[j].iFreq);
         printf("         top: %.0f\n", fTop);
         printf("      bottom: %.0f\n", fBottom);
         delay(200);
      }
   }
}
if(fBottom != 0)
{
   fSub = (fTop / (pClsTmplt->iNumRecs * fBottom));
}
pScore->sub.fTop = fTop;
pScore->sub.fBottom = fBottom;
pScore->sub.iNumRecs = pClsTmplt->iNumRecs;
pScore->sub.fResult = fSub;

/* Caculate score for the originating organization fields */
iMatch = FALSE;
fTop = 0;
fBottom = 0;
for(i = 0; (pRecTmplt->pOrg[i].szKwrd[0] != '\0'); i++)
{
   iMatch = FALSE;
   for(j = 0; (pClsTmplt->pOrg[j].szKwrd[0] != '\0' &&
      iMatch != TRUE); j++)
   {
      clrscr();
      printf("Template %d - Originating Org\n", pScore->iDBFNum);
      printf("  Record: %s \n", pRecTmplt->pOrg[i].szKwrd);
      printf("Template: %s \n", pClsTmplt->pOrg[j].szKwrd);
      delay(5);

      if(strcmp(pRecTmplt->pOrg[i].szKwrd,
            pClsTmplt->pOrg[j].szKwrd) == 0)
      {
         iMatch = TRUE;
```

```
            fTop += ((pRecTmplt->pOrg[i].iFreq) *
               (pClsTmplt->pOrg[j].iFreq));
            fBottom += pRecTmplt->pOrg[i].iFreq;
            printf("record freq: %d\n", pRecTmplt->pOrg[i].iFreq);
            printf(" class freq: %d\n", pClsTmplt->pOrg[j].iFreq);
            printf("        top: %.0f\n", fTop);
            printf("     bottom: %.0f\n", fBottom);
            delay(200);
         }
      }
   }
   if(fBottom != 0)
   {
      fOrg = (fTop / (pClsTmplt->iNumRecs * fBottom));
   }
   pScore->org.fTop = fTop;
   pScore->org.fBottom = fBottom;
   pScore->org.iNumRecs = pClsTmplt->iNumRecs;
   pScore->org.fResult = fOrg;

   /* Caculate score for the record type fields */
   iMatch = FALSE;
   fTop = 0;
   fBottom = 0;
   for(i = 0; (pRecTmplt->pTyp[i].szKwrd[0] != '\0'); i++)
   {
      iMatch = FALSE;
      for(j = 0; (pClsTmplt->pTyp[j].szKwrd[0] != '\0' &&
         iMatch != TRUE); j++)
      {
         clrscr();
         printf("Template %d - Record Type\n", pScore->iDBFNum);
         printf("  Record: %s \n", pRecTmplt->pTyp[i].szKwrd);
         printf("Template: %s \n", pClsTmplt->pTyp[j].szKwrd);
         delay(5);

         if(strcmp(pRecTmplt->pTyp[i].szKwrd,
                pClsTmplt->pTyp[j].szKwrd) == 0)
         {
            iMatch = TRUE;
            fTop += ((pRecTmplt->pTyp[i].iFreq) *
               (pClsTmplt->pTyp[j].iFreq));
            fBottom += pRecTmplt->pTyp[i].iFreq;
            printf("record freq: %d\n", pRecTmplt->pTyp[i].iFreq);
            printf(" class freq: %d\n", pClsTmplt->pTyp[j].iFreq);
            printf("        top: %.0f\n", fTop);
            printf("     bottom: %.0f\n", fBottom);
            delay(200);
         }
      }
   }
   if(fBottom != 0)
   {
      fTyp = (fTop / (pClsTmplt->iNumRecs * fBottom));
   }
   pScore->typ.fTop = fTop;
   pScore->typ.fBottom = fBottom;
   pScore->typ.iNumRecs = pClsTmplt->iNumRecs;
   pScore->typ.fResult = fTyp;

   /* Caculate score for the media type fields */
   iMatch = FALSE;
   fTop = 0;
   fBottom = 0;
   for(i = 0; (pRecTmplt->pMed[i].szKwrd[0] != '\0'); i++)
```

```
{
   iMatch = FALSE;
   for(j = 0; (pClsTmplt->pMed[j].szKwrd[0] != '\0' &&
      iMatch != TRUE); j++)
   {
      clrscr();
      printf("Template %d - Media Type\n", pScore->iDBFNum);
      printf("  Record: %s \n", pRecTmplt->pMed[i].szKwrd);
      printf("Template: %s \n", pClsTmplt->pMed[j].szKwrd);
      delay(5);

      if(strcmp(pRecTmplt->pMed[i].szKwrd,
            pClsTmplt->pMed[j].szKwrd) == 0)
      {
         iMatch = TRUE;
         fTop += ((pRecTmplt->pMed[i].iFreq) *
            (pClsTmplt->pMed[j].iFreq));
         fBottom += pRecTmplt->pMed[i].iFreq;
         printf("record freq: %d\n", pRecTmplt->pMed[i].iFreq);
         printf(" class freq: %d\n", pClsTmplt->pMed[j].iFreq);
         printf("        top: %.0f\n", fTop);
         printf("     bottom: %.0f\n", fBottom);
         delay(200);
      }
   }
}
if(fBottom != 0)
{
   fMed = (fTop / (pClsTmplt->iNumRecs * fBottom));
}
pScore->med.fTop = fTop;
pScore->med.fBottom = fBottom;
pScore->med.iNumRecs = pClsTmplt->iNumRecs;
pScore->med.fResult = fMed;

/* Caculate score for the record format fields */
iMatch = FALSE;
fTop = 0;
fBottom = 0;
for(i = 0; (pRecTmplt->pFrm[i].szKwrd[0] != '\0'); i++)
{
   iMatch = FALSE;
   for(j = 0; (pClsTmplt->pFrm[j].szKwrd[0] != '\0' &&
      iMatch != TRUE); j++)
   {
      clrscr();
      printf("Template %d - Record Format\n", pScore->iDBFNum);
      printf("  Record: %s \n", pRecTmplt->pFrm[i].szKwrd);
      printf("Template: %s \n", pClsTmplt->pFrm[j].szKwrd);
      delay(5);

      if(strcmp(pRecTmplt->pFrm[i].szKwrd,
            pClsTmplt->pFrm[j].szKwrd) == 0)
      {
         iMatch = TRUE;
         fTop += ((pRecTmplt->pFrm[i].iFreq) *
            (pClsTmplt->pFrm[j].iFreq));
         fBottom += pRecTmplt->pFrm[i].iFreq;
         printf("record freq: %d\n", pRecTmplt->pFrm[i].iFreq);
         printf(" class freq: %d\n", pClsTmplt->pFrm[j].iFreq);
         printf("        top: %.0f\n", fTop);
         printf("     bottom: %.0f\n", fBottom);
         delay(200);
      }
   }
```

```
    }
    if(fBottom != 0)
    {
       fFrm = (fTop / (pClsTmplt->iNumRecs * fBottom));
    }
    pScore->frm.fTop = fTop;
    pScore->frm.fBottom = fBottom;
    pScore->frm.iNumRecs = pClsTmplt->iNumRecs;
    pScore->frm.fResult = fFrm;


    /* Calculate the composite score for the document */
    fScore[0] = (fSub * .3) + (fOrg * .2) + (fTyp * .3) +
          (fMed * .1) + (fFrm * .1);
    fScore[1] = (fSub * .2) + (fOrg * .2) + (fTyp * .2) +
          (fMed * .2) + (fFrm * .2);
    fScore[2] = (fSub * .5) + (fOrg * .3) + (fTyp * .0) +
          (fMed * .1) + (fFrm * .1);

    clrscr();
    printf("30/20/30/10/10: %.3f\n", fScore[0]);
    printf("20/20/20/20/20: %.3f\n", fScore[1]);
    printf("50/30/00/10/10: %.3f\n", fScore[2]);
    delay(500);

    pScore->fScore[0] = fScore[0];
    pScore->fScore[1] = fScore[1];
    pScore->fScore[2] = fScore[2];

}
```

```
/*-----------------------------------------------------------------------
PROJECT: racs.prj

FILE: stemmer.c

PURPOSE:
  This module is an implementation of the Porter suffix stripping algorithm.
  This code was written by C. Fox, 1990.

FUNCTIONS:
  char *stem(register char *word)
  static int WordSize(register char *word)
  static int ContainsVowel(register char *word)
  static int EndsWithCVC(register char *word)
  static int AddAnE(register char *word)
  static int RemoveAnE(register char *word)
  static int ReplaceEnd(register char *word, struct RULELIST *rule)
-----------------------------------------------------------------------*/

#define EXTERN extern
#include "racs.h"

#define EOS '\0'
#define IsVowel(c) \
   (('a'==(c))||('e'==(c))||('i'==(c))||('o'==(c))||('u'==(c)))

struct RULELIST {
   int id;
   char *old_end;
   char *new_end;
   int old_offset;
   int new_offset;
   int min_root_size;
   int (*condition)();
};

static char LAMBDA[1] = "";
static char *end;

static struct RULELIST step1a_rules[] =
{
   101,  "sses",  "ss",      3,  1, -1,  NULL,
   102,  "ies",   "i",       2,  0, -1,  NULL,
   103,  "ss",    "ss",      1,  1, -1,  NULL,
   104,  "s",     LAMBDA,    0, -1, -1,  NULL,
   000,  NULL,    NULL,      0,  0,  0,  NULL,
};

static struct RULELIST step1b_rules[ ] =
{
   105,  "eed",   "ee",      2,  1,  0,  NULL,
   106,  "ed",    LAMBDA,    1, -1, -1,  ContainsVowel,
   107,  "ing",   LAMBDA,    2, -1, -1,  ContainsVowel,
   000,  NULL,    NULL,      0,  0,  0,  NULL,
};

static struct RULELIST step1b1_rules[] =
{
   108,  "at",    "ate",     1,  2, -1,  NULL,
   109,  "bl",    "ble",     1,  2, -1,  NULL,
   110,  "iz",    "ize",     1,  2, -1,  NULL,
   111,  "bb",    "b",       1,  0, -1,  NULL,
   112,  "dd",    "d",       1,  0, -1,  NULL,
   113,  "ff",    "f",       1,  0, -1,  NULL,
   114,  "gg",    "g",       1,  0, -1,  NULL,
```

```
   115,  "mm",     "m",       1,  0, -1,  NULL,
   116,  "nn",     "n",       1,  0, -1,  NULL,
   117,  "pp",     "p",       1,  0, -1,  NULL,
   118,  "rr",     "r",       1,  0, -1,  NULL,
   119,  "tt",     "t",       1,  0, -1,  NULL,
   120,  "ww",     "w",       1,  0, -1,  NULL,
   121,  "xx",     "x",       1,  0, -1,  NULL,
   122,  LAMBDA,   "e",      -1,  0, -1,  AddAnE,
   000,  NULL,     NULL,      0,  0,  0,  NULL,
};

static struct RULELIST step1c_rules[] =
{
   123,  "y",      "i",       0,  0, -1,  ContainsVowel,
   000,  NULL,     NULL,      0,  0,  0,  NULL,
};

static struct RULELIST step2_rules[] =
{
   203, "ational", "ate",    6,  2,  0,  NULL,
   204, "tional",  "tion",   5,  3,  0,  NULL,
   205, "enci",    "ence",   3,  3,  0,  NULL,
   206, "anci",    "ance",   3,  3,  0,  NULL,
   207, "izer",    "ize",    3,  2,  0,  NULL,
   208, "abli",    "able",   3,  3,  0,  NULL,
   209, "alli",    "al",     3,  1,  0,  NULL,
   210, "entli",   "ent",    4,  2,  0,  NULL,
   211, "eli",     "e",      2,  0,  0,  NULL,
   213, "ousli",   "ous",    4,  2,  0,  NULL,
   214, "ization", "ize",    6,  2,  0,  NULL,
   215, "ation",   "ate",    4,  2,  0,  NULL,
   216, "ator",    "ate",    3,  2,  0,  NULL,
   217, "alism",   "al",     4,  1,  0,  NULL,
   218, "iveness", "ive",    6,  2,  0,  NULL,
   219, "fulnes",  "ful",    5,  2,  0,  NULL,
   220, "ousness", "ous",    6,  2,  0,  NULL,
   221, "aliti",   "al",     4,  1,  0,  NULL,
   222, "iviti",   "ive",    4,  2,  0,  NULL,
   223, "biliti",  "ble",    5,  2,  0,  NULL,
   000, NULL,      NULL,     0,  0,  0,  NULL,
};

static struct RULELIST step3_rules[] =
{
   301,  "icate",  "ic",     4,  1,  0,  NULL,
   302,  "ative",  LAMBDA,   4, -1,  0,  NULL,
   303,  "alize",  "al",     4,  1,  0,  NULL,
   304,  "iciti",  "ic",     4,  1,  0,  NULL,
   305,  "ical",   "ic",     3,  1,  0,  NULL,
   308,  "ful",    LAMBDA,   2, -1,  0,  NULL,
   309,  "ness",   LAMBDA,   3, -1,  0,  NULL,
   000,  NULL,     NULL,     0,  0,  0,  NULL,
};

static struct RULELIST step4_rules[] =
{
   401,  "al",     LAMBDA,   1, -1,  1,  NULL,
   402,  "ance",   LAMBDA,   3, -1,  1,  NULL,
   403,  "ence",   LAMBDA,   3, -1,  1,  NULL,
   405,  "er",     LAMBDA,   1, -1,  1,  NULL,
   406,  "ic",     LAMBDA,   1, -1,  1,  NULL,
   407,  "able",   LAMBDA,   3, -1,  1,  NULL,
   408,  "ible",   LAMBDA,   3, -1,  1,  NULL,
   409,  "ant",    LAMBDA,   2, -1,  1,  NULL,
   410,  "ement",  LAMBDA,   4, -1,  1,  NULL,
```

```
   411,  "ment",    LAMBDA,  3, -1,  1,  NULL,
   412,  "ent",     LAMBDA,  2, -1,  1,  NULL,
   423,  "sion",    "s",     3,  0,  1,  NULL,
   424,  "tion",    "t",     3,  0,  1,  NULL,
   415,  "ou",      LAMBDA,  1, -1,  1,  NULL,
   416,  "ism",     LAMBDA,  2, -1,  1,  NULL,
   417,  "ate",     LAMBDA,  2, -1,  1,  NULL,
   418,  "iti",     LAMBDA,  2, -1,  1,  NULL,
   419,  "out",     LAMBDA,  2, -1,  1,  NULL,
   420,  "ive",     LAMBDA,  2, -1,  1,  NULL,
   421,  "ize",     LAMBDA,  2, -1,  1,  NULL,
   000,  NULL,      NULL,    0,  0,  0,  NULL,
};

static struct RULELIST step5a_rules[] =
{
   501,  "e",       LAMBDA,  0, -1,  1,  NULL,
   502,  "e",       LAMBDA,  0, -1, -1,  RemoveAnE,
   000,  NULL,      NULL,    0,  0,  0,  NULL,
};

static struct RULELIST step5b_rules[] =
{
   503,  "ll",      "l",     1,  0,  1,  NULL,
   000,  NULL,      NULL,    0,  0,  0,  NULL,
};


char *stem(register char *word)
{
   int rule;

   for(end = word; *end != EOS; end++)
   {
      if(!isalpha(*end))
      {
         return(FALSE);
      }
   }
   end--;

   ReplaceEnd(word, step1a_rules);
   rule = ReplaceEnd(word, step1b_rules);
   if((106==rule) || ( 107 == rule))
   {
      ReplaceEnd(word, step1b1_rules);
   }
   ReplaceEnd( word, step1c_rules);
   ReplaceEnd( word, step2_rules);
   ReplaceEnd( word, step3_rules);
   ReplaceEnd( word, step4_rules);
   ReplaceEnd( word, step5a_rules);
   ReplaceEnd( word, step5b_rules);

   return(word);
}


static int WordSize(register char *word)
{
   register int result;
   register int state;

   result = 0;
   state = 0;
```

```
   while (EOS != *word)
   {
      switch(state)
      {
         case 0:
            state = (IsVowel(*word)) ? 1 : 2;
            break;

         case 1:
            state = (IsVowel(*word)) ? 1 : 2;
            if ( 2 == state ) result++;
            break;

         case 2:
            state = (IsVowel(*word) || ('y'== *word)) ? 1 : 2;
            break;
      }
      word++;
   }
   return( result);
} /* WordSize */


static int ContainsVowel(register char *word)
{
   if ( EOS == *word)
   {
      return(FALSE);
   }
   else
   {
      return (IsVowel(*word) || ( NULL != strpbrk(word+1,"aeiouy")));
   }
} /* ContainsVowel */


static int EndsWithCVC(register char *word)
{
   int length;

   if ((length = strlen(word)) < 2)
   {
      return(FALSE);
   }
   else
   {
      end = word + length -1;
      return((NULL == strchr("aeiouwxy", *end--))
         && (NULL != strchr("aeiouy",    *end--))
         && (NULL == strchr("aeiou",     *end)));
   }
} /* EndsWithCVC */


static int AddAnE(register char *word)
{
   return((1 == WordSize(word)) &&  EndsWithCVC(word));
} /* AddAnE */


static int RemoveAnE(register char *word)
{
   return(( 1 == WordSize(word)) &&  !EndsWithCVC(word));
} /* RemoveAnE */
```

```
static int ReplaceEnd(register char *word, struct RULELIST *rule)
{
   register char *ending;
   char tmp_ch;

   while(0 != rule->id)
   {
      ending = end - rule->old_offset;
      if(word != ending)
      {
         if(0 == strcmp(ending, rule->old_end))
         {
            tmp_ch = *ending;
            *ending = EOS;
            if(rule->min_root_size < WordSize(word))
            {
               if(!rule->condition || (*rule->condition) (word))
               {
                  (void) strcat( word, rule->new_end);
                  end = ending + rule->new_offset;
                  return( rule->id);
               }
            }
            *ending = tmp_ch;
            return( rule->id);
         }
      }
      rule++;
   }
   return(rule->id);
} /* ReplaceEnd */
```

```
/*-----------------------------------------------------------------------------
PROJECT: racs.prj

FILE: stoplist.c

PURPOSE:
  This module contains functions to work with a stop list.

FUNCTIONS:
  int loadStoplist(char *szStoplist[])
  void unloadStoplist(char *szStoplist[], int iNumWords)
  int checkStoplist(char *szTerm, char *szStoplist[], int iNumWords)
-----------------------------------------------------------------------------*/

#define EXTERN extern
#include "racs.h"

int loadStoplist(char *szStoplist[])
{
   char szBuff[21];
   char szBuffOld[21];
   char szHeader[20] = "stoplist";

   int i;

   FILE *fpStoplist;

   if((fpStoplist = fopen(szpGetConfig(szHeader, 1), "r")) == NULL)
   {
      displayError("opening stoplist");
   }

   i = 0;
   szBuff[0] = NULL;
   do
   {
      strcpy(szBuffOld, szBuff);
      fscanf(fpStoplist, "%s", szBuff);
      if((szStoplist[i] = (char *)malloc(strlen(szBuff)+1)) == NULL)
      {
         displayError("allocating memory for stoplist");
      }
      if(strcmp(szBuffOld, szBuff) != 0)
      {
         strcpy(szStoplist[i], szBuff);
         i++;
      }
   }while(strcmp(szBuffOld, szBuff) != 0);

   fclose(fpStoplist);
   return i;
}


void unloadStoplist(char *szStoplist[], int iNumWords)
{
   int i;

   for(i = 0; i < iNumWords; i++)
   {
      free(szStoplist[i]);
   }
}
```

```
int checkStoplist(char *szTerm, char *szStoplist[], int iNumWords)
{
   int i;

   for(i = 0; i < iNumWords; i++)
   {
      if(strcmp(szTerm, szStoplist[i]) == 0)
      {
         return TRUE;
      }
   }
   return FALSE;
}
```

```
/*-------------------------------------------------------------------------
PROJECT: racs.prj

FILE: template.c

PURPOSE:
  This module contains all of the functions for manipulating the five class
  templates which represent the 5 databases.

FUNCTIONS:
  int genCLASSTMPLT(int iDBFNum)
  void addToCLASSTMPLT(char *szTerm, struct CLASSTMPLT *pTmplt, int iField)
  void logCLASSTMPLT(struct CLASSTMPLT *pTmplt, int iDBFNum)
-------------------------------------------------------------------------*/

#define EXTERN extern
#include "racs.h"


int genCLASSTMPLT(int iDBFNum)
{
   FILE *fpCurDBF;
   FILE *fpClsTmplt;
   char szHeader1[20] = "dbfile";
   char szHeader2[20] = "template";

   char szBuff[255];
   char szTermBuff[51];
   char *pString, *pToken;
   char szToken[41];
   int iTokenType;
   int iField;

   char *szStoplist[500];
   int iNumWords;
   int iMatch;

   int i, j, k;

   struct DB3HEADER db3header;

   struct CLASSTMPLT tmplt;
   struct CLASSTMPLT *pTmplt;
   struct DB3RECORD db3record;
   pTmplt = &tmplt;

   iNumWords = loadStoplist(szStoplist);

   memset(&tmplt, 0, sizeof(struct CLASSTMPLT));

   if((fpCurDBF = fopen(szpGetConfig(szHeader1, iDBFNum), "rb")) == NULL)
   {
      displayError("could not open database file");
   }

   if((fread(&db3header, sizeof(struct DB3HEADER), 1, fpCurDBF)) == NULL)
   {
      displayError("read error (database header)");
   }

   tmplt.iNumRecs = db3header.lNumberRecords;

   fseek(fpCurDBF, db3header.nFirstRecordOffset, SEEK_SET);

   for(i = 0; i < db3header.lNumberRecords; i++)
```

```
{
   memset(&db3record, 0, sizeof(struct DB3RECORD));

   if((fread(&db3record, sizeof(struct DB3RECORD), 1, fpCurDBF)) == NULL)
   {
      displayError("read error (database record)");
   }

   if(db3record.szStatus[0] != '*')
   {
      for(j = 0; j < 5; j++)
      {
         memset(&szBuff, '\0', sizeof(szBuff));

         switch(j)
         {
            case 0:
               strncpy(szBuff, db3record.szSubject, 255);
               szBuff[254] = '\0';
               iField = 's';
               break;

            case 1:
               strncpy(szBuff, db3record.szOriginOrg, 101);
               iField = 'o';
               break;

            case 2:
               strncpy(szBuff, db3record.szRecType, 51);
               iField = 't';
               break;

            case 3:
               strncpy(szBuff, db3record.szMediaType, 51);
               iField = 'm';
               break;

            case 4:
               strncpy(szBuff, db3record.szRecFormat, 51);
               iField = 'f';
               break;
         }

         pString = szBuff;

         iTokenType = UNDEFINED;

         while(iTokenType != EOL && iTokenType != LEXERROR)
         {
            pToken = szToken;

            getTerms(&pString, pToken, &iTokenType);

            if(iTokenType != EOL)
            {
               if(iTokenType == ALLALPHA)
               {
                  strcpy(szTermBuff, strlwr(szToken));

                  iMatch = checkStoplist(szTermBuff,
                        szStoplist, iNumWords);

                  if(iMatch != TRUE)
                  {
                     strcpy(szTermBuff, stem(strlwr(szToken)));
```

```
                        addToCLASSTMPLT(szTermBuff, pTmplt, iField);
                     }
                  }

                  if(iTokenType == NONWORD)
                  {
                     strcpy(szTermBuff, strlwr(szToken));

                     iMatch = checkStoplist(szTermBuff,
                           szStoplist, iNumWords);

                     if(iMatch != TRUE)
                     {
                        strcpy(szTermBuff, strlwr(szToken));
                        addToCLASSTMPLT(szTermBuff, pTmplt, iField);
                     }
                  }
               }
            }
         }
      }
   }
   unloadStoplist(szStoplist, iNumWords);

   fclose(fpCurDBF);

   if((fpClsTmplt = fopen(szpGetConfig(szHeader2, iDBFNum), "wb")) == NULL)
   {
      displayError("could not create template file");
   }

   if((fwrite((char *)&tmplt, sizeof(tmplt), 1, fpClsTmplt)) != 1)
   {
      displayError("write error (class template)");
   }
   fclose(fpClsTmplt);

   logCLASSTMPLT(pTmplt, iDBFNum);

   return 1;
}


void addToCLASSTMPLT(char *szTerm, struct CLASSTMPLT *pTmplt, int iField)
{
   int i;

   if(iField == 's')
   {
      for(i = 0; pTmplt->pSub[i].szKwrd[0] != '\0'; i++)
      {
         if(strcmp(pTmplt->pSub[i].szKwrd, szTerm) == 0)
         {
            pTmplt->pSub[i].iFreq += 1;
            return;
         }
      }
      strcpy(pTmplt->pSub[i].szKwrd, szTerm);
      pTmplt->pSub[i].iFreq = 1;
   }

   if(iField == 'o')
   {
      for(i = 0; pTmplt->pOrg[i].szKwrd[0] != '\0'; i++)
      {
```

```
            if(strcmp(pTmplt->pOrg[i].szKwrd, szTerm) == 0)
            {
               pTmplt->pOrg[i].iFreq += 1;
               return;
            }
         }
         strcpy(pTmplt->pOrg[i].szKwrd, szTerm);
         pTmplt->pOrg[i].iFreq = 1;
      }

      if(iField == 't')
      {
         for(i = 0; pTmplt->pTyp[i].szKwrd[0] != '\0'; i++)
         {
            if(strcmp(pTmplt->pTyp[i].szKwrd, szTerm) == 0)
            {
               pTmplt->pTyp[i].iFreq += 1;
               return;
            }
         }
         strcpy(pTmplt->pTyp[i].szKwrd, szTerm);
         pTmplt->pTyp[i].iFreq = 1;
      }

      if(iField == 'm')
      {
         for(i = 0; pTmplt->pMed[i].szKwrd[0] != '\0'; i++)
         {
            if(strcmp(pTmplt->pMed[i].szKwrd, szTerm) == 0)
            {
               pTmplt->pMed[i].iFreq += 1;
               return;
            }
         }
         strcpy(pTmplt->pMed[i].szKwrd, szTerm);
         pTmplt->pMed[i].iFreq = 1;
      }

      if(iField == 'f')
      {
         for(i = 0; pTmplt->pFrm[i].szKwrd[0] != '\0'; i++)
         {
            if(strcmp(pTmplt->pFrm[i].szKwrd, szTerm) == 0)
            {
               pTmplt->pFrm[i].iFreq += 1;
               return;
            }
         }
         strcpy(pTmplt->pFrm[i].szKwrd, szTerm);
         pTmplt->pFrm[i].iFreq = 1;
      }
   }


   void logCLASSTMPLT(struct CLASSTMPLT *pTmplt, int iDBFNum)
   {
      FILE *fpLogFile;
      char szHeader[20] = "templatetxt";

      int i, j;

      if((fpLogFile = fopen(szpGetConfig(szHeader, iDBFNum), "w")) == NULL)
      {
         displayError("opening log file");
      }
```

```
   fprintf(fpLogFile, "---- CLASS TEMPLATE DATABASE %d -----------\n",
        iDBFNum);
   fprintf(fpLogFile, "NUMBER OF RECORDS: %d\n", pTmplt->iNumRecs);
   for(i = 0; i < 5; i++)
   {
      if(i == 0)
      {
         fprintf(fpLogFile, "SUBJECT:\n");
         for(j = 0; pTmplt->pSub[j].szKwrd[0] != '\0'; j++)
         {
            fprintf(fpLogFile, "Kwrd %-5d%-18sFreq = %d\n", j,
                 pTmplt->pSub[j].szKwrd, pTmplt->pSub[j].iFreq);
         }
         fprintf(fpLogFile, "\n");
      }

      if(i == 1)
      {
         fprintf(fpLogFile, "ORIGINATING ORGANIZATION:\n");
         for(j = 0; pTmplt->pOrg[j].szKwrd[0] != '\0'; j++)
         {
            fprintf(fpLogFile, "Kwrd %-5d%-18sFreq = %d\n", j,
                 pTmplt->pOrg[j].szKwrd, pTmplt->pOrg[j].iFreq);
         }
         fprintf(fpLogFile, "\n");
      }

      if(i == 2)
      {
         fprintf(fpLogFile, "RECORD TYPE:\n");
         for(j = 0; pTmplt->pTyp[j].szKwrd[0] != '\0'; j++)
         {
            fprintf(fpLogFile, "Kwrd %-5d%-18sFreq = %d\n", j,
                 pTmplt->pTyp[j].szKwrd, pTmplt->pTyp[j].iFreq);
         }
         fprintf(fpLogFile, "\n");
      }

      if(i == 3)
      {
         fprintf(fpLogFile, "MEDIA TYPE:\n");
         for(j = 0; pTmplt->pMed[j].szKwrd[0] != '\0'; j++)
         {
            fprintf(fpLogFile, "Kwrd %-5d%-18sFreq = %d\n", j,
                 pTmplt->pMed[j].szKwrd, pTmplt->pMed[j].iFreq);
         }
         fprintf(fpLogFile, "\n");
      }

      if(i == 4)
      {
         fprintf(fpLogFile, "RECORD FORMAT:\n");
         for(j = 0; pTmplt->pFrm[j].szKwrd[0] != '\0'; j++)
         {
            fprintf(fpLogFile, "Kwrd %-5d%-18sFreq = %d\n", j,
                 pTmplt->pFrm[j].szKwrd, pTmplt->pFrm[j].iFreq);
         }
         fprintf(fpLogFile, "\n");
      }
   }
   fclose(fpLogFile);
}
```

Appendix D: RACS Configuration File

```
[STOPLIST]
STOPLIST.TXT

[TEMPLATE]
CAT1.TPL
CAT2.TPL
CAT3.TPL
CAT4.TPL
CAT5.TPL

[DBFILE]
CAT1.DBF
CAT2.DBF
CAT3.DBF
CAT4.DBF
CAT5.DBF
TEMP.DBF

[DBFBACKUP]
CAT1.DBB
CAT2.DBB
CAT3.DBB
CAT4.DBB
CAT5.DBB

[TEMPLATE]
CAT1.TPL
CAT2.TPL
CAT3.TPL
CAT4.TPL
CAT5.TPL

[TEMPLATETXT]
CAT1TPL.TXT
CAT2TPL.TXT
CAT3TPL.TXT
CAT4TPL.TXT
CAT5TPL.TXT

[LOGFILE]
LOGFILE.TXT
SCORELOG.TXT

[LOGBACKUP]
LOGFILE.BAK
SCORELOG.BAK

[END]
```

Appendix E: RACS Stop List

a
about
above
across
af
after
again
against
all
almost
alone
along
already
also
although
always
among
an
and
another
any
anybody
anyone
anything
anywhere
are
area
areas
around
as
ask
asked
asking
asks
at
away
b
back
backed
backing
backs
be
because
became
become
becomes
been
before
began
behind
being
beings
best
better
between
big
both
but
by
c
came
can
cannot
case
cases
certain
certainly
clear
clearly
come
could
d
did
differ
different
differently
do
does
done
down
downed
downing
downs
during
e
each
early
either
end
ended
ending
ends
enough
even
evenly
ever
every
everybody
everyone
everything
everywhere
f
face
faces
fact
facts
far
felt
few
find
finds
first
for
form
four
from
full
fully
further
furthered
furthering
furthers
g
gave
general
generally
get
gets
give
given
gives
go
going
good
goods
got
great
greater
greatest
group
grouped
grouping
groups
h
had
has
have
having
he
her
herself
here
high
higher
highest
him
himself
his
how
however
i
if
important
in
interest
interested
interesting
interests
into
is
it
its
itself
j
just
k
keep
keeps
kind
knew
know
known
knows
l
large
largely
last
later
latest
least
less
let
lets
like
likely
long
longer
longest
m
made
make
making
man
many
may
me
member
members
men
might
more
most
mostly
mr
mrs
much
must
my
myself
n
necessary
need
needed
needing
needs
never
new
newer
newest
next
no
non
not
nobody
noone
nothing

now
nowhere
number
numbered
numbering
numbers
o
of
off
often
old
older
oldest
on
once
one
only
open
opened
opening
opens
or
order
ordered
ordering
orders
other
others
our
out
over
p
part
parted
parting
parts
per
perhaps
place
places
point
pointed
pointing
points
possible
present
presented
presenting
presents
problem
problems
put
puts
q
quite
r
rather
really
right
room
rooms
s
said
same
saw
say
says
second
seconds
see
seem
seemed
seeming
seems
sees
several
shall
she
should
show
showed
showing
shows
side
sides
since
small
smaller
smallest
so
some
somebody
someone
something
somewhere
state
states
still
such
sure
t
take
taken
than
that
the
their
them
then
there
therefore
these
they
thing
things
think
thinks
this
those
though
thought
thoughts
three
through
thus
to
today
together
too
took
toward
turn
turned
turning
turns
two
u
under
until
up
upon
us
use
uses
used
v
very
w
want
wanted
wanting
wants
was
way
ways
we
well
wells
went
were
what
when
where
whether
which
while
who
whole
whose
why
will
with
within
without
work
worked
working
works
would
x
y
year
years
yet
you
young
younger
youngest
your
yours
z

Appendix F: Excerpt from logfile.txt

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
****************************** INPUT ANALYSIS ******************************
SUBJECT:
Request                 ALLALPHA    request
for                     ALLALPHA    -SW-
Evaluation              ALLALPHA    evalu
,                       PUNCT       ----
WP                      NONWORD     wp
-                       PUNCT       ----
960264                  NONWORD     960264
Voluntary               ALLALPHA    voluntari
Reduction               ALLALPHA    reduct
in                      ALLALPHA    -SW-
the                     ALLALPHA    -SW-
Federal                 ALLALPHA    feder
Workforce               ALLALPHA    workforc

ORIGINATING ORGANIZATION:
ASC                     NONWORD     asc
/                       PUNCT       ----
MOS                     NONWORD     mos

RECORD TYPE:
official                ALLALPHA    offici
memorandum              ALLALPHA    memorandum

MEDIA TYPE:
paper                   ALLALPHA    paper

RECORD FORMAT:
paper                   ALLALPHA    paper


***************************** RECORD TEMPLATE ******************************
SUBJECT:
Kwrd 0    request           Freq = 1
Kwrd 1    evalu             Freq = 1
Kwrd 2    wp                Freq = 1
Kwrd 3    960264            Freq = 1
Kwrd 4    voluntari         Freq = 1
Kwrd 5    reduct            Freq = 1
Kwrd 6    feder             Freq = 1
Kwrd 7    workforc          Freq = 1

ORIGINATING ORGANIZATION:
Kwrd 0    asc               Freq = 1
Kwrd 1    mos               Freq = 1

RECORD TYPE:
Kwrd 0    offici            Freq = 1
Kwrd 1    memorandum        Freq = 1

MEDIA TYPE:
Kwrd 0    paper             Freq = 1

RECORD FORMAT:
Kwrd 0    paper             Freq = 1
```

```
***************************** SCORING RESULTS ******************************
30/20/30/10/10          20/20/20/20/20          50/30/00/10/10
DBF RANK SCORE          DBF RANK SCORE          DBF RANK SCORE
 1   3   0.529           1   3   0.629          1   3   0.314
 2   4   0.511           2   4   0.611          2   2   0.356
 3   5   0.200           3   5   0.400          3   5   0.200
 4   2   0.571           4   2   0.657          4   3   0.314
 5   1   0.605           5   1   0.686          5   1   0.629

1 SUB  2 / (14 *  1) = 0.143 (0.3 = 0.043) (0.2 = 0.029) (0.5 = 0.071)
1 ORG  2 / (14 *  1) = 0.143 (0.2 = 0.029) (0.2 = 0.029) (0.3 = 0.043)
1 TYP 24 / (14 *  2) = 0.857 (0.3 = 0.257) (0.2 = 0.171) (0.0 = 0.000)
1 MED 14 / (14 *  1) = 1.000 (0.1 = 0.100) (0.2 = 0.200) (0.1 = 0.100)
1 FRM 14 / (14 *  1) = 1.000 (0.1 = 0.100) (0.2 = 0.200) (0.1 = 0.100)
                           Totals: 0.529         0.629         0.314

2 SUB  5 / (18 *  1) = 0.278 (0.3 = 0.083) (0.2 = 0.056) (0.5 = 0.139)
2 ORG  1 / (18 *  1) = 0.056 (0.2 = 0.011) (0.2 = 0.011) (0.3 = 0.017)
2 TYP 26 / (18 *  2) = 0.722 (0.3 = 0.217) (0.2 = 0.144) (0.0 = 0.000)
2 MED 18 / (18 *  1) = 1.000 (0.1 = 0.100) (0.2 = 0.200) (0.1 = 0.100)
2 FRM 18 / (18 *  1) = 1.000 (0.1 = 0.100) (0.2 = 0.200) (0.1 = 0.100)
                           Totals: 0.511         0.611         0.356

3 SUB  0 / ( 3 *  0) = 0.000 (0.3 = 0.000) (0.2 = 0.000) (0.5 = 0.000)
3 ORG  0 / ( 3 *  0) = 0.000 (0.2 = 0.000) (0.2 = 0.000) (0.3 = 0.000)
3 TYP  0 / ( 3 *  0) = 0.000 (0.3 = 0.000) (0.2 = 0.000) (0.0 = 0.000)
3 MED  3 / ( 3 *  1) = 1.000 (0.1 = 0.100) (0.2 = 0.200) (0.1 = 0.100)
3 FRM  3 / ( 3 *  1) = 1.000 (0.1 = 0.100) (0.2 = 0.200) (0.1 = 0.100)
                           Totals: 0.200         0.400         0.200

4 SUB  2 / ( 7 *  2) = 0.143 (0.3 = 0.043) (0.2 = 0.029) (0.5 = 0.071)
4 ORG  1 / ( 7 *  1) = 0.143 (0.2 = 0.029) (0.2 = 0.029) (0.3 = 0.043)
4 TYP 14 / ( 7 *  2) = 1.000 (0.3 = 0.300) (0.2 = 0.200) (0.0 = 0.000)
4 MED  7 / ( 7 *  1) = 1.000 (0.1 = 0.100) (0.2 = 0.200) (0.1 = 0.100)
4 FRM  7 / ( 7 *  1) = 1.000 (0.1 = 0.100) (0.2 = 0.200) (0.1 = 0.100)
                           Totals: 0.571         0.657         0.314

5 SUB 45 / (21 *  3) = 0.714 (0.3 = 0.214) (0.2 = 0.143) (0.5 = 0.357)
5 ORG 10 / (21 *  2) = 0.238 (0.2 = 0.048) (0.2 = 0.048) (0.3 = 0.071)
5 TYP 20 / (21 *  2) = 0.476 (0.3 = 0.143) (0.2 = 0.095) (0.0 = 0.000)
5 MED 21 / (21 *  1) = 1.000 (0.1 = 0.100) (0.2 = 0.200) (0.1 = 0.100)
5 FRM 21 / (21 *  1) = 1.000 (0.1 = 0.100) (0.2 = 0.200) (0.1 = 0.100)
                           Totals: 0.605         0.686         0.629

Correct DBF: 5  Offsets: 0  0  0
```

Appendix G: Excerpt from scorelog.txt

```
DBF: 3  Offsets: 4   4   4
DBF: 2  Offsets: 4   4   4
DBF: 5  Offsets: 4   4   4
DBF: 2  Offsets: 1   1   1
DBF: 1  Offsets: 4   4   4
DBF: 4  Offsets: 4   4   4
DBF: 1  Offsets: 0   0   0
DBF: 5  Offsets: 0   2   0
DBF: 2  Offsets: 4   4   4
DBF: 2  Offsets: 2   2   2
DBF: 1  Offsets: 0   0   1
DBF: 2  Offsets: 1   1   2
DBF: 2  Offsets: 1   1   2
DBF: 5  Offsets: 0   0   0
DBF: 1  Offsets: 2   3   4
DBF: 5  Offsets: 0   0   0
DBF: 2  Offsets: 1   1   2
DBF: 2  Offsets: 1   1   1
DBF: 5  Offsets: 2   2   0
DBF: 1  Offsets: 0   0   0
DBF: 2  Offsets: 1   1   2
DBF: 2  Offsets: 2   2   3
DBF: 2  Offsets: 1   1   1
DBF: 1  Offsets: 0   0   1
DBF: 5  Offsets: 0   0   1
DBF: 5  Offsets: 0   0   0
DBF: 5  Offsets: 0   0   0
DBF: 2  Offsets: 1   1   0
DBF: 1  Offsets: 1   1   1
DBF: 5  Offsets: 0   0   0
DBF: 2  Offsets: 1   1   0
DBF: 5  Offsets: 0   0   0
DBF: 5  Offsets: 0   0   0
DBF: 5  Offsets: 0   0   0
DBF: 1  Offsets: 0   0   1
DBF: 4  Offsets: 2   2   4
DBF: 2  Offsets: 2   2   4
DBF: 2  Offsets: 2   2   4
DBF: 2  Offsets: 1   1   0
DBF: 5  Offsets: 0   0   0
DBF: 4  Offsets: 0   0   0
DBF: 1  Offsets: 0   0   2
DBF: 1  Offsets: 0   1   0
DBF: 4  Offsets: 0   0   0
DBF: 2  Offsets: 1   1   1
DBF: 2  Offsets: 1   1   1
DBF: 4  Offsets: 0   0   0
DBF: 3  Offsets: 0   0   0
DBF: 5  Offsets: 0   0   0
DBF: 1  Offsets: 0   0   0
DBF: 5  Offsets: 0   0   0
DBF: 5  Offsets: 2   1   0
DBF: 2  Offsets: 0   0   0
DBF: 2  Offsets: 3   3   3
DBF: 2  Offsets: 3   3   3
```

Appendix H: Sample Class Template

```
---- CLASS TEMPLATE DATABASE 4 -----------
NUMBER OF RECORDS: 13
SUBJECT:
Kwrd 0    lesson          Freq = 1
Kwrd 1    learn           Freq = 1
Kwrd 2    oper            Freq = 1
Kwrd 3    desert          Freq = 1
Kwrd 4    storm           Freq = 1
Kwrd 5    freez           Freq = 1
Kwrd 6    munition        Freq = 1
Kwrd 7    custodi         Freq = 1
Kwrd 8    account         Freq = 1
Kwrd 9    custom          Freq = 1
Kwrd 10   satisfact       Freq = 1
Kwrd 11   survei          Freq = 2
Kwrd 12   audit           Freq = 2
Kwrd 13   inspect         Freq = 6
Kwrd 14   report          Freq = 2
Kwrd 15   semiannu        Freq = 1
Kwrd 16   self            Freq = 4
Kwrd 17   manag           Freq = 4
Kwrd 18   comment         Freq = 1
Kwrd 19   44595           Freq = 1
Kwrd 20   xxx             Freq = 1
Kwrd 21   weapon          Freq = 1
Kwrd 22   aeronaut        Freq = 1
Kwrd 23   system          Freq = 1
Kwrd 24   center          Freq = 1
Kwrd 25   asc             Freq = 1
Kwrd 26   wpafb           Freq = 2
Kwrd 27   oh              Freq = 1
Kwrd 28   45433           Freq = 1
Kwrd 29   battle          Freq = 1
Kwrd 30   staff           Freq = 2
Kwrd 31   support         Freq = 1
Kwrd 32   air             Freq = 1
Kwrd 33   force           Freq = 1
Kwrd 34   agenc           Freq = 1
Kwrd 35   afia            Freq = 1
Kwrd 36   function        Freq = 1
Kwrd 37   review          Freq = 3
Kwrd 38   fmr             Freq = 1
Kwrd 39   wing            Freq = 1
Kwrd 40   level           Freq = 1
Kwrd 41   logist          Freq = 1
Kwrd 42   plan            Freq = 1
Kwrd 43   organiz         Freq = 1
Kwrd 44   structur        Freq = 1
Kwrd 45   unit            Freq = 1
Kwrd 46   program         Freq = 2
Kwrd 47   refer           Freq = 1
Kwrd 48   sup             Freq = 1
Kwrd 49   1               Freq = 1
Kwrd 50   afr             Freq = 1
Kwrd 51   123-1           Freq = 1
Kwrd 52   semi            Freq = 1
Kwrd 53   annual          Freq = 2
Kwrd 54   follow          Freq = 1
Kwrd 55   statement       Freq = 1
```

```
Kwrd 56    requir              Freq = 1
Kwrd 57    feder               Freq = 1
Kwrd 58    financi             Freq = 1
Kwrd 59    integr              Freq = 1
Kwrd 60    act                 Freq = 1
Kwrd 61    fmfia               Freq = 1
Kwrd 62    1982                Freq = 1
Kwrd 63    aflc                Freq = 1
Kwrd 64    special             Freq = 1
Kwrd 65    item                Freq = 1
Kwrd 66    91-3                Freq = 1
Kwrd 67    continu             Freq = 1
Kwrd 68    evalu               Freq = 1
Kwrd 69    personnel           Freq = 1

ORIGINATING ORGANIZATION:
Kwrd 0     2750                Freq = 7
Kwrd 1     abw                 Freq = 2
Kwrd 2     ck                  Freq = 1
Kwrd 3     sptg                Freq = 3
Kwrd 4     cce                 Freq = 2
Kwrd 5     fmc                 Freq = 1
Kwrd 6     ms                  Freq = 3
Kwrd 7     88                  Freq = 1
Kwrd 8     cc                  Freq = 2
Kwrd 9     mss                 Freq = 4
Kwrd 10    msi                 Freq = 1
Kwrd 11    asc                 Freq = 1
Kwrd 12    ig                  Freq = 1
Kwrd 13    none                Freq = 1
Kwrd 14    cvx                 Freq = 1

RECORD TYPE:
Kwrd 0     offici              Freq = 12
Kwrd 1     memorandum          Freq = 12
Kwrd 2     2519                Freq = 1

MEDIA TYPE:
Kwrd 0     paper               Freq = 13

RECORD FORMAT:
Kwrd 0     paper               Freq = 13
```

Appendix I: Common Tables and Rules

The tables and rules listed in the column labeled OLD represent the designations for the rules in AFR 4-20 Vol 2. The tables and rules listed in the column labeled NEW are the new designations found for the same rules in AFMAN 37-139. The rules are ordered in relation to their AFR 4-20 Vol 2 designations.

NEW		OLD		
TABLE	RULE	TABLE	RULE	DESCRIPTION
37-3	3	4-3	3	dispatch and delivery receipts on accountable mail
37-3	14	4-3	14	accountable container receipts
37-6	1	4-6	1	publications/forms requisitions and requirements
37-6	7	4-6	7	publication bulletins
37-7	7	5-1	7	operating instructions record copies - at MAJCOM and above
37-7	8	5-1	8	operating instructions record copies - below MAJCOM
37-7	9	5-1	9	base bulletins
37-11	2	10-1	2	general correspondence - temporary
37-11	4	10-1	4	transitory material
37-11	5	10-1	5	reading file
37-11	6	10-1	6	message file (extra copies of messages)
37-11	10	10-1	10	office projects/studies (background and working materials)
37-11	12	10-1	12	staff meetings and conferences (not covered elsewhere) - at MAJCOM and above
37-11	13	10-1	13	staff meetings and conferences (not covered elsewhere) - below MAJCOM
37-12	5	10-2	5	suspense control
37-13	1	10-3	1	background material to orders in rules 2, 2.1 and 4
37-13	2	10-3	3	temporary orders (M, P, T, Y, PA and PB series)

NEW		OLD		
TABLE	RULE	TABLE	RULE	DESCRIPTION
37-14	1	11-1	1	office administrative files
37-14	4	11-1	4	project control and support (working papers, transcribed steno notes or tapes)
37-14	6	11-1	6	reports, controlled and uncontrolled - not covered elsewhere
37-14	8	11-1	8	reports, controlled and uncontrolled - information copies
37-14	9	11-1	9	precedent files
37-14	10	11-1	10	office instructions, additional duty handbooks/workbooks
37-14	11	11-1	11	building or office services (not covered elsewhere)
37-14	12	11-1	12	presentation aids (not covered elsewhere)
37-14	14	11-1	14	general reference publications
37-14	15	11-1	15	technical/specialized reference materials
37-14	17	11-1	17	organizational planning - at directorate level or above
37-14	18	11-1	18	organizational planning - below directorate level
90-4	2	11-2	2	congressional inquiries - below HQ USAF
37-15	7	11-2	12	host-tenant support agreements
37-15	9	11-2	12.2	other support agreements
37-15	13	11-2	15	GAO audit reports - below HQ USAF
37-15	14	11-2	16	official visits/staff visits (offices performing visits)
37-15	15	11-2	17	official visits/staff visits (offices visited)
37-15	16	11-2	18	official visits/staff visits (intermediate, monitoring or evaluating offices)
37-15	17	11-2	19	official visits/staff visits (visits notifications, itineraries)
37-15	18	11-2	20	official visits/staff visits (visit schedules)
37-15	19	11-2	21	delegations/designations of authority and additional duty assignments
37-15	27	11-2	29	locator or personal data
37-15	31	11-2	33	internal inspections/self-inspection checklists/Inventories (not covered elsewhere)
37-15	32	11-2	34	Overtime requests (for disposition instructions see T177-21, R03 or T176-03, R39.01)
37-18	18	11-5	18	Word Processing Files (floppy disks or hard drives containing letters, memos, messages, reports)

NEW		OLD		
TABLE	RULE	TABLE	RULE	DESCRIPTION
37-19	2	12-1	2	files maintenance and disposition (AF Form 80)
37-19	3	12-1	3	retirement, transfer/shipment records (SF 135 and SF 258) - at initiator's office for records placed in staging area
37-19	4	12-1	4	retirement, transfer/shipment records (SF 135 and SF 258) - at office of record manager (RM) for records placed in staging areas
37-19	5	12-1	5	retirement, transfer/shipment records (SF 135 and SF 258) - records retired to records centers
37-19	6	12-1	6	retirement, transfer/shipment records (SF 135 and SF 258) - transferred records
37-19	23	12-1	23	Freedom of Information Act (FOIA) Program - correspondence relating to administering FOIA
37-19	24	12-1	24	Freedom of Information Act (FOIA) Program - correspondence responding to requests
37-19	26	12-1	26	Freedom of Information Act (FOIA) Program - denials not appealed
38-2	11	25-2	11	productivity enhancement
38-3	11	26-1	11	manpower authorization - machine listing derived from the manpower authorization file
38-3	17	26-1	17	manpower change requests - approved/disapproved requests at MAJCOMs
38-3	18	26-1	18	manpower change requests - approved/disapproved requests below MAJCOMs
38-3	18.1	26-1	18.1	manpower change requests - information copies kept for monitoring purposes
36-4	14	30-4	14	RIP products
36-12	2	35-1	2	personnel information file
36-15	16	35-4	16	individual job descriptions
36-15	27	35-4	27	military sponsor program - at losing activity
36-15	28	35-4	28	military sponsor program - at gaining activity
36-29	4	40-4	4	performance/incentive awards
36-29	10	40-4	10	leave transfer/sharing programs (submitted or resulting from a request/contribution of leave)
36-29	11	40-4	11	leave transfer/sharing programs (background info)
36-30	7	40-5	7	position management
——	--	40-8	13	supervisor's employee work folder - documents filed by the supervisor in the work folder
36-38	6	50-2	6	unit training program
13-10	31	60-5	31	flight evaluation folders
23-11	40	67-11	40	equipment custodian file

NEW		OLD		
TABLE	RULE	TABLE	RULE	DESCRIPTION
64-1	14	70-1	7.3	contractor general files - duplicate/working
63-6	3	74-1	3	surveillance records
37-13	1	75-3	15	area clearance for oversea theaters
61-1	19	80-1	19	scientific and technical reference files
90-2	1	123-1	1	inspection reports not otherwise covered in this table - at MAJCOMs
90-2	7	123-1	7	inspection reports not otherwise covered in this table - background material
177-1	8	177-1	8	reports of accounting and finance activities
177-21	3	177-21	3	individual attendance and overtime
177-21	12	177-21	12	payroll control registers
177-21	19	177-21	19	listings (not covered by rules 1 through 18)
177-32	30	177-32	30	unit leave control log - unit copy
177-32	30.1	177-32	30.1	unit leave control log - MPSMA copy
31-4	9	205-1	9	security control records - SF 700, 701, 702
31-4	15	205-1	15	access control records
31-4	22	205-1	22	record suspense receipt and destruction certificate file for secret material - inactive records
31-4	26	205-1	26	security termination statements - at unit of assignment
31-4	29	205-1	29	security termination statements - at unit of assignment for civilian personnel
31-10	4	207-1	4	records of visitors - requests for visits to restricted areas
31-10	5	207-1	5	records of visitors - authorization for contractors to visit in connection with classified matters
84-1	7	210-1	7	source documents - for history reports
33-9	9	700-9	9	telephone toll calls - AF Form 1072
63-9	5	800-1	5	system acquisition program files - at system program offices
63-9	6	800-1	6	system acquisition program files - at monitoring, supporting, testing and participating activities
63-9	12	800-1	12	memorandums of agreement (MOAs)
36-33	2	900-1	2	special honors, trophies and awards - initiating activities
36-33	3	900-1	3	special honors, trophies and awards - nonselected nominations

NEW		OLD		
TABLE	RULE	TABLE	RULE	DESCRIPTION
36-33	16	900-1	15	favorable communications
36-33	17	900-1	16	outstanding personnel programs, e.g., outstanding NCO/Airman award, Junior Officer of the Quarter, outstanding Manager of the Year, AFA representative
36-34	1	900-2	1	suggestions, inventions and scientific achievements - at suggestion program office
36-34	2	900-2	2	suggestions, inventions and scientific achievements - at evaluating offices

Appendix J. 88 SPTG/CCE Files Maintenance and Disposition Plan

ITEM	TITLE	LOCATION	DISPOSITION
1	FILE MAINT & DISPOSITION PLAN, CTRL RECORD LABEL AND RELATED RCRD	FRONT OF FILES & EACH SERIES	T 12-01 R02.00
2	READING FILE		T 10-01 R05.00
3	DELEGATION/DESIGNATIONS OF AUTHORITY & ADDITIONAL DUTY ASSIGN		T 11-02 R21.00
4	TRANSITORY MATERIAL		T 10-01 R04.00
	4-1 JAN-APR-JUL-OCT		
	4-2 FEB-MAY-AUG-NOV		
	4-3 MAR-JUN-SEP-DEC		
5	GENERAL CORRESPONDENCE (TEMPORARY)		T 10-01 R02.00
6	OFFICE ADMINISTRATIVE FILE - INTERNAL ADMIN OR HOUSEKEEPING		T 11-01 R01.00
	6-1 SECURITY		
	6-2 DISASTER PREPAREDNESS		
	6-3 INSTALLATION MANAGEMENT		
	6-3-1 FACILITIES		
	6-3-2 SUPPLIES/EQUIPMENT		
	6-4 SAFETY		
	6-5 ADMINISTRATION OF OFFICE PERSONNEL		
7	ADMINISTRATIVE SUPPORT COMMITTEE & BOARD RECORDS		T 25-03 R07.00
	7-1 FINANCIAL WORKING GROUP (FWG)		
	7-2 FINANCIAL MANAGEMENT BOARD (FMB)		
	7-3 EEO ADVISORY COMMITTEE		
	7-4 OCCUPATIONAL SAFETY, FIRE PREVENTION & HEALTH COMMITTEE		
8	HOST-TENANT SUPPORT AGREEMENTS		T 11-02 R12.00
9	OFFICE PROJECTS/STUDIES BELOW MAJ SUB COMD - NO PUBLICATION ISSUED		T 10-01 R09.00
10	SUPERVISOR'S EMPLOYEE WORK FOLDER	NCOIC, ADMIN'S DESK	T 40-08 R13.00
11	WORD PROCESSING FILES		T 11-05 R18.00
12	INTERNAL INSPECTIONS/SELF-INSP CHECK LISTS/INVENTORIES		T 11-02 R33.00
13	OFFICIAL VISITS/STAFF VISITS - AT OFFICES OR ORGANIZATIONS VISITED		T11-02 R 17.00
14	INSPECTION REPORTS - AT INSPECTED ACTIVITIES, MONIT/EVAL/APPR AUTH		T123-01 R03.00
15	SUGS, INVENTIONS, & SCIENTIFIC ACHIEVEMENTS - AT EVAL OFF		T900-02 R02.00
16	GENERAL TRAINING REPORTS	GENERAL TRAINING REPORTS	T 50-01 R19.00
17	FUNDING RECORDS - PROGRAM PROJECT AND APPROPRIATE CONTROL		T172-03 R04.00
	17-1 FINANCIAL PLAN		
18	SPECIAL HONORS, TROPHIES, AWARDS -		T900-01 R02.00

	AT INITIATING ACTIVITIES		
19	OUTSTANDING PERSONNEL PROGRAMS		T900-01 R16.00
20	TEMP. ORDERS (M-, P-, T-, Y-, PA-, PB-, SPECIAL, & SQD NON-PREFIXED		T 10-03 R03.00
	20-1 GF SERIES ORDERS		
	20-2 GA SERIES ORDERS		
	20-3 M SERIES ORDERS		
	20-4 TRAVEL ORDERS		
21	SUSPENSE CONTROL (FILE COPIES OR EXTRA COPIES TO MANAGE FLOW)		T 10-02 R05.00
22	PRECEDENT FILES - EXTRA COPIES OF PRECEDENT FILES SELECTED RECORDS		T 11-01 R09.00
23	PAYROLL CONTROL REGISTER DOCUMENT FILES	PAYROLL CONTROL REGISTERS	T177-21 R12.00
	23-1 WORK SCHEDULES/CHANGES		
	23-2 FORMAT II TIMESHEETS		

Appendix K: Sample Records

The tables included in this appendix contain the data collected on each record used in this thesis.

DBF	RULE	TO	FROM	SUBJECT	AUTHOR	DATE	RECORD TYPE	MEDIA TYPE	RECORD FORMAT
1	T11-02 R 21	none	none	Delegation/Designations of Authority and Additional Duty Assignments	none	none	no specific format	paper	paper
1	T11-02 R 21	88 CS/SCMNPE	88 SPTG/CC	Appointment/Change of ADPE Equipment Custodian	Commander	none	official memorandum	paper	paper
1	T11-02 R 21	88 SUPS/LGSME	88 SPTG/CC	Appointment/Change of Equipment Custodian	Commander	none	official memorandum	paper	paper
1	T11-02 R 21	88 CG/IMAAAC	88 ABW/CCE	Authorization for Receipt of Registered/Certified Mail	Assistant to the Executive	8-Oct-96	official memorandum	paper	paper
1	T11-02 R 21	88 SUPS/LGSMSR	88 SPTG/CC	Appointment of DRMO Withdrawal Custodians	Commander	15-Aug-96	official memorandum	paper	paper
1	T11-02 R 21	MSS/DPMR	88 SPTG/CC	Unit Personnel Records Group Check-Out	Commander	30-Jul-96	official memorandum	paper	paper
1	T11-02 R 21	88 CG/IMADA	88 SPTG/CCE	Appointment of Functional Area Records Manager (FARM)	88 SPTG/CCE	23-Jul-96	official memorandum	paper	paper
1	T11-02 R 21	none	88 SPTG/CCE	Request for and Record of Customer Account Representative Designation	NCOIC, Information Management for Support Group	19-Jul-96	AF Form 1846	paper	paper
1	T11-02 R 21	88 CS/SCMNPE	88 SPTG/CC	Appointment/Change of Equipment Custodian	Commander	18-Jul-96	official memorandum	paper	paper
1	T11-02 R 21	WPAFB SECURITY MANAGERS	ASC/SYSI	Request for Security Manager Email Addresses	Chief, Security Planning and Integration	9-Jul-96	official memorandum	paper	paper
1	T11-02 R 21	88 CG/SCXP	88 SPTG/CD	Telephone Service Officer	Deputy Commander	26-Jun-96	official memorandum	paper	paper
1	T11-02 R 21	88 CG/IMADF	88 SPTG/CD	Appointment of Freedom of Information Act (FOIA) Monitors	Deputy Commander	25-Jun-96	official memorandum	paper	paper
1	T11-02 R 21	88 CG/IMAAAC	88 ABW/CCE	Authorization for Receipt of Registered/Certified Mail	Assistant to the Executive	24-Jun-96	official memorandum	paper	paper
1	T11-02 R 21	88 SPTG/CCE	88 SPTG/CC	Appointment of Orders Approving Officials	Commander	24-Jun-96	official memorandum	paper	paper
1	T11-02 R 21	88 CCSG/SCOCA	88 SPTG/CD	Message Receipt Authorizations	Deputy Commander	20-Jun-96	official memorandum	paper	paper
1	T11-02 R 21	MSS/DPMPR	88 SPTG/CD	Unit Personnel Records Group Check-Out	Deputy Commander	18-Jun-96	official memorandum	paper	paper
1	T11-02 R 21	ASC/MOS	88 SPTG/CD	Suggestion Program Monitor	Deputy Commander	18-Jun-96	official memorandum	paper	paper
1	T11-02 R 21	88 CEG/CEXD	88 SPTG/CD	Disaster Control Group Representatives	Deputy Commander	3-Apr-96	official memorandum	paper	paper
1	T11-02 R 21	88 CG/IMAA	88 SPTG	FW: Request Update to Point of Contact (POC) Information for Office Symbol/Title Changes and Management Roster System Input	Jane M. Doe	15-Sep-95	email	paper	paper
1	T11-02 R 21	88 ABW/SPTG	ASC/CVH	Functional/SPO Focal Point Listing - Update	Director, Integrated Workforce Management Office	12-Sep-95	official memorandum	paper	paper
1	T11-02 R 21	Greene County Board of Commissioners	88 SPTG/CC	Letter to Greene County Board of Commissioners designating new member of PIC	Commander	11-May-95	personal letter	paper	paper
1	T11-02 R 21	88 SPTG/CC	ASC/CVH	Focal Points for Management Operations	Chief, Integrated Workforce Management Office	2-May-95	official memorandum	paper	paper

DBF	RULE	TO	FROM	SUBJECT	AUTHOR	DATE	RECORD TYPE	MEDIA TYPE	RECORD FORMAT
1	T11-02 R 21	DANTES	88 SPTG/CC	Appointment of DANTES Alternate Test Control Officer (ATCO), ID#1627	Commander	30-Mar-95	official memorandum	paper	paper
1	T11-02 R 21	ASC/ASIS	88 SPTG/CC	Authorization to Request Personnel Security Actions	Commander	17-Feb-95	official memorandum	paper	paper
1	T11-02 R 21	DISTRIBUTION LIST	88 ABW/XPR	Point of Contact for WPAFB Plans	Chief, War/Contingency Plans Office	6-Dec-94	official memorandum	paper	paper
1	T11-02 R 21	ACFQ	DP	AFR 177-16, Administrative Control of Appropriations	Deputy Commander for Support	17-Sep-90	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	DISTRIBUTION LIST	88 CG/SCCV	Procurement of Printing and Duplicating Products	Chief, Multimedia/Visual Information Branch	none	official memorandum	paper	paper
2	T 11-1 R1 6-3-1	none	none	Supply End of Year Procedures	none	none	no specific format	paper	paper
2	T 11-1 R1 6-3-2	88 SPTG/CCN	88 ABW/FMFSLV	Review/Deletions of PFMRs	Material & Voucher Control	20-Nov-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	DMATS-Dayton	88 ABW/CCE	DMATS-DAYTON Communications Service Requirement - Request for purchase of one COMDIAL telephone	88 ABW/CCE	26-Sep-96	DMATS-D Form 1070	paper	paper
2	T 11-1 R1 6-3-2	TSO 88 SPTG/CC	88 CG/SCXP	DMATS Communication Service Request Status	Communication Management Officer	9-Sep-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	88 SPTG/CCN	88 CG/SCXRB	Telephone Bill Verification	Ch, Financial Mgt Section	5-Sep-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	88 SPTG/CCE	88 CS/SCMT	Distribution of the DMATS-Dayton Telephone Directory	Manager, DMATS-Dayton	4-Sep-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	88 SPTG/CCN	88 CG/SCXRB	Telephone Bill Verification	Ch, Financial Mgt Section	1-Aug-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	88 MSS/CC 88 SPS/CC 88 SPTG/SV 88 SPTG/DPC	88 SPTG/CD	Freeze on Equipment Purchases Lifted	Deputy Commander	1-Aug-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	TSO 88 SPTG/CC	88 CG/SCXP	DMATS Communication Service Request Status	Communication Management Officer	26-Jun-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	DMATS-DAYTON	88 SPTG/CC	DMATS-DAYTON Communications Service Requirement - Request to have operation of five telephones changed	88 SPTG/CC	18-Jun-96	DMATS-D Form 1070	paper	paper
2	T 11-1 R1 6-3-2	none	88 SPTG/CC	Financial Liability Investigation of Property Loss - for loss of personal pager	Commander	11-Jun-96	DD Form 200	paper	paper
2	T 11-1 R1 6-3-2	88 SPTG/CCN	88 CG/SCXRB	Telephone Bill Verification	Ch, Financial Mgt Section	3-Jun-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	88 CG/SCXP	DMATS-D/SCMT	Status of DMATS-Dayton Communications Service Requirement	Manager, DMATS-Dayton	10-May-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	DMATS-D/SCMTOD	88 SPTG/CCE	Information Updates	NCOIC, Administration	10-May-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	88 SPTG/CCN	88 CG/SCXRB	Telephone Bill Verification	Ch, Financial Mgt Section	9-May-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	88 SPTG/CCE	88 ABW/FMC	Report of Survey	Assistant ROS Monitor for Equipment	3-May-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	88 SPTG/CCN	88 CG/SCXRB	Telephone Bill Verification	Ch, Financial Mgt Section	22-Apr-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	88 CG/CA	88 SPTG/CCE	Annual Verification of Equipment Custodians	NCOIC, Administrative Support	11-Apr-96	official memorandum	paper	paper

DBF	RULE	TO	FROM	SUBJECT	AUTHOR	DATE	RECORD TYPE	MEDIA TYPE	RECORD FORMAT
2	T 11-1 R1 6-3-2	88 ABW/CCE	DMATS-D/SCMT	Status of DMATS-Dayton Communications Service Requirement	Manager, DMATS-Dayton	11-Jan-96	official memorandum	paper	paper
2	T 11-1 R1 6-3-1	88 SPTG/CCE	88 CG/SCCC	Request for a Cellular Telephone	Customer Service Center Representative	13-Dec-95	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	DISTRIBUTION LIST	ASC/CV	Unit Account Representatives	Vice Commander	13-Nov-95	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	DISTRIBUTION LIST	HQ AFMC/SVP	Subsistence Prime Vendor (SPV)	Chief, Programs	20-Oct-95	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	DMATS-DAYTON	88 ABW/CCE	DMATS-DAYTON Communications Service Requirement - request for purchase of one COMDIAL telephone	88 ABW/CCE	26-Sep-95	DMATS-D Form 1070	paper	paper
2	T 11-1 R1 6-3-2	DISTRIBUTION LIST	88 CG/CC	Use of the International Merchant Purchase Authorization Card (IMPAC) to Acquire Communications and Computer Resources	Commander	6-Sep-95	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	88 CG/SCXF	88 SPTG/CCX	Fund Cite Authorization	88 SPTG Plans & Programs	31-Aug-95	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	DMATS-DAYTON	88 SPTG	DMATS-DAYTON Communications Service Requirement - request for purchase of three COMDIAL telephones	88 ABW/CCE	25-Aug-95	DMATS-D Form 1070	paper	paper
2	T 11-1 R1 6-3-2	2A, 2B, 2C, 2D	ASC/CC	Payment of American Express (AMEX) Bills	Commander	4-Aug-95	official memorandum	paper	paper
2	T 11-1 R1 6-3-1	88 MSS	88 CG/SCXP	Telephone Bill Verification	Communications Management Officer	24-May-95	official memorandum	paper	paper
2	T 11-1 R1 6-3-2	88 MSS/CCQ	88 MSS/DPE	American Express (AmEx) Government Card	Director, Education & Training	12-May-95	official memorandum	paper	paper
3	T 11-1 R1 6-4	none	none	Employee Safety and Health Record - Jane M. Doe	88 SPTG/CCE	none	AF Form 55	paper	paper
3	T 11-1 R1 6-4	none	none	Employee Safety and Health Record - John B. Good	645 SPTG/CCXC	none	AF Form 55	paper	paper
3	T 11-1 R1 6-4	none	none	Employee Safety and Health Record - Bill R. Smith	88 SPTG/CC	none	AF Form 55	paper	paper
3	T 11-1 R1 6-4	none	none	Employee Safety and Health Record - David J. Jones	88 SPTG/CCE	none	AF Form 55	paper	paper
3	T 11-1 R1 6-4	none	none	Employee Safety and Health Record - Peter P. Piper	88 SPTG/CCE	none	AF Form 55	paper	paper
4	T 11-2 R33	ASC/MO	SPTG/CCE	Customer Satisfaction Survey	Support Group Administration	26-Oct-95	official memorandum	paper	paper
4	T 11-2 R33	88 SPS/CC	SPTG/CCE	Freezing of Munitions Custody Account	NCOIC, Support Group Administration	19-Sep-95	official memorandum indorsement	paper	paper
4	T 11-2 R33	88 ABW/CC 88 ABW/XP 88 LG/CC	ASC/IG	Air Force Inspection Agency (AFIA) - Functional Management Review (FMR) on Wing-Level Logistics Plans Organizational Structure	Inspector General Office	21-Jul-95	official memorandum	paper	paper
4	T 11-2 R33	ASC/IG	88 SPTG/CC	Management Comments to Report of Audit 44595XXX, Management of Weapons, Aeronautical System Center (ASC), Wright-Patterson AFB OH 45433 (Project No. 95445028)	Commander	13-Jul-95	official memorandum	paper	paper
4	T 11-2 R33	2750 MSS/MS	FMC	Audit/Inspection Reports and Surveys	Base Audit Focal Point	10-Mar-92	official memorandum	paper	paper
4	T 11-2 R33	2750 ABW/CC	2750 MSS/CC	Annual Statement Required Under the Federal Managers' Financial Integrity Act (FMFIA) of 1982	Commander	22-Aug-91	official memorandum	paper	paper
4	T 11-2 R33	HQ AFLC/CK	2750 ABW/CK	Lessons Learned - Operation Desert Storm	Air Force Reserve Advisor	12-Jun-91	official memorandum	paper	paper

DBF	RULE	TO	FROM	SUBJECT	AUTHOR	DATE	RECORD TYPE	MEDIA TYPE	RECORD FORMAT
4	T 11-2 R33	2750 ABW/CVX	2750 MS	Semiannual Self-Inspection	NCOIC, Mission Support Squadron	31-May-91	official memorandum	paper	paper
4	T 11-2 R33	MSU	2750 ABW/CVX	Review of Self-Inspection	Inspection Program Monitor	31-May-91	official memorandum	paper	paper
4	T 11-2 R33	2750 MSS/MS	2750 MSS/MSI	Battle Staff/Support Staff	Chief, Base Information Management	29-May-91	official memorandum	paper	paper
4	T 11-2 R33	none	none	Unit Self-Inspection Program Review for Reference WPAFB Sup 1 to AFR 123-1	CVX	13-May-91	AF Form 2519	paper	paper
4	T 11-2 R33	2750 ABW/CVX	2750 MSS/MS	Semi-Annual Self Inspection Follow-Up	NCOIC, Mission Support Squadron	13-Mar-91	official memorandum	paper	paper
4	T 11-2 R33	2750 SPS/SPAI	2750 MSS/MS	AFLC Special Interest Item 91-3, Continuing Evaluation of Personnel Program	NCOIC, Mission Support Squadron	13-Feb-91	official memorandum	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPS/SPA	Suggestion Evaluation and Transmittal -- Position Sensitivity WP-960286	Chief, Admin & Reports Flight	6-Sep-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	88 MSS/CCE	ASC/MOS	Request for Evaluation, WP-960282 Safety Review of Contractor's Facility	Suggestion Program Assistant	29-Aug-96	official memorandum	paper	paper
5	T 900-2 R2	ASC/MOS	88 MSS/DPM	Suggestion Evaluation and Transmittal -- Updating Personal Information WP-960288	Chief, Military Personnel Flight	29-Aug-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	ASC/MOS	AFMEA/PLDP	Air Force Suggestion WP 960002, Develop One Form for Designation of Beneficiaries	Management Assistant	26-Aug-96	official memorandum	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPTG/DPC	Suggestion Evaluation and Transmittal -- Rif Procedures WP-960280	Chief Civilian Personnel Division	23-Aug-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	88 SPS/CCE	ASC/MOS	Request for Evaluation, WP-960289 Stop or Yield Sign at Exit of AMCII	Suggestion Program Manager	15-Aug-96	official memorandum	paper	paper
5	T 900-2 R2	88 MSS/CCE	ASC/MOS	Request for Evaluation, WP-960286 Position Sensitivity	Suggestion Program Manager	15-Aug-96	official memorandum	paper	paper
5	T 900-2 R2	88 MSS/CCE	ASC/MOS	Request for Evaluation, WP-960283 Credit for Unused Civilian Sick Leave	Suggestion Program Manager	9-Aug-96	official memorandum	paper	paper
5	T 900-2 R2	HQ AFMC/DPE	88 MSS/DPE	Suggestion Evaluation and Transmittal -- Acquisition Professional Development Courses WP-960275	Acting Chief, Education and Training	8-Aug-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	88 SPS/CCE	ASC/MOS	Request for Evaluation, WP-960276 Parking Lot Telephones	Suggestion Program Manager	5-Aug-96	official memorandum	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPTG/DPC	Suggestion Evaluation and Transmittal -- Electronic Submission of Leave Request WP-960243	Deputy Chief Civilian Personnel Division	5-Aug-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	88 MSS/CCE	ASC/MOS	Request for Evaluation, WP-960267 Job Swap/Lateral Transfer Facilitator	Suggestion Program Manager	31-Jul-96	official memorandum	paper	paper
5	T 900-2 R2	88 MSS/CCE	ASC/MOS	Request for Evaluation, WP-960265 Placement Referral System	Suggestion Program Manager	31-Jul-96	official memorandum	paper	paper
5	T 900-2 R2	88 MSS/CCE	ASC/MOS	Request for Evaluation, WP-960264 Voluntary Reduction in the Federal Workforce	Suggestion Program Manager	31-Jul-96	official memorandum	paper	paper
5	T 900-2 R2	88 SPTG/CC	ASC/MOS	Request for Evaluation, WP-960251 Provide Recycling Containers	Suggestion Program Manager	24-Jul-96	official memorandum	paper	paper
5	T 900-2 R2	88 SPTG/CC	ASC/MOS	Request for Evaluation, WP-960245 Recycling	Suggestion Program Manager	17-Jul-96	official memorandum	paper	paper
5	T 900-2 R2	88 SPS/CCE	ASC/MOS	Request for Evaluation, WP-960228 Traffic Control at Area B I 675 Gate	Suggestion Program Manager	1-Jul-96	official memorandum	paper	paper

DBF	RULE	TO	FROM	SUBJECT	AUTHOR	DATE	RECORD TYPE	MEDIA TYPE	RECORD FORMAT
5	T 900-2 R2	88 MSS/CCE	ASC/MOS	Request for Evaluation, WP-960226 Change Civilian Personnel WWW Page	Suggestion Program Manager	1-Jul-96	official memorandum	paper	paper
5	T 900-2 R2	ASC/MOS	88 MSS/DPM	Suggestion Evaluation and Transmittal -- Electronic Personnel Information WP-960206	Chief, Military Personnel Flight	14-Jun-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPTG/CCE	Request for Evaluation, WP-960187 Change Civilian Personnel WWW Page	NCOIC, Administration	30-May-96	official memorandum	paper	paper
5	T 900-2 R2	ASC/MOS	12 FTW/MO	Implementation Documents for Suggestion WP 92-0191 - Additional Qualifying Experience Added to Career Brief	Suggestion Program Assistant	21-May-96	official memorandum	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPTG/CCE	Projected Implementation Requirements For Suggestion, WP-960070, Recycle Bins at Super Playground	NCOIC, Administrations	20-May-96	official memorandum	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPTG/DPCPC	Implementation of Suggestion WP-960101	Chief Plans & Requirements Flight	10-May-96	official memorandum	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPTG/SV	Suggestion Evaluation and Transmittal -- Sale Scrap Paper WP-960142	Recreation Flight Chief	18-Apr-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	88 SPS/CCE	ASC/MOS	Request for Evaluation, WP-960136A Parking Blocks/Signs	Suggestion Program Assistant	9-Apr-96	official memorandum	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPTG/DPC	Suggestion Evaluation and Transmittal -- Eliminate Time Cards for GS Employees WP-960155	Acting Chief Civilian Personnel Division	5-Apr-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPTG/SVA	Suggestion Evaluation and Transmittal -- Safety WP-960106	Chief, Military Support Flight	1-Apr-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPS/SPAIS	Suggestion Evaluation and Transmittal -- Small Arms Training WP-960092	Commander	28-Mar-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPS/SPAIS	Suggestion Evaluation and Transmittal -- AFLC Form 1499, Quarters Check Program WP-960071	Commander	22-Mar-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	HQ AFMC/DPEP	88 MSS/DPEOM	Suggestion Evaluation and Transmittal -- Training Requirements WP-960131	Chief, Education and Training	19-Mar-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	ASC/MOS	88 MSS/DPM	Suggestion Evaluation and Transmittal -- Military 401(k) Plan WP-960123	Commander, 88 Mission Support Squadron	8-Mar-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPTG/SV	Suggestion Evaluation and Transmittal -- Alcohol Regulations on Base WP-960103	Chief, Membership Support Flight	28-Feb-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	HQ AFMC/DPXD	88 MSS/DPMDS	Suggestion Evaluation and Transmittal -- Automate Personnel/Base Locating Functions WP-960095	Chief Military Personnel Flight	20-Feb-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	ASC/MOS	HQ AFPC/DPPR	Suggestion Evaluation and Transmittal -- Wear of Air National Guard Ribbons WP-960077	Chief, Recognition Programs Branch	20-Feb-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	HQ 11 WG/XPM	AFDPO	Suggestion Evaluation and Transmittal -- Management of AF Form 707A WP-960073	Chief Support Division	16-Feb-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPS	Suggestion Evaluation and Transmittal -- Bloodborne Pathogen Danger with Leather Police Gear WP-960102	Commander, 88 SPS	15-Feb-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPTG/CCE	Suggestion Evaluation and Transmittal -- Classified Briefing Cost Avoidance WP-960015	88 SPTG/CCE	8-Feb-96	AF Form 1000-1	paper	paper
5	T 900-2 R2	88 SPS/CCE	ASC/MOS	Additional Information Requested for Suggestion No. WP 960075, Safety	Suggestion Program Assistant	24-Jan-96	official memorandum	paper	paper
5	T 900-2 R2	ASC/MOS	88 SPS	Suggestion Evaluation and Transmittal -- WPAFB Form 1499, Quarters Check Report WP-960071	Chief, Security Police	23-Jan-96	AF Form 1000-1	paper	paper

Appendix L: Offset Data

Exp Weight 0.15

SAMPLE 1

WHOLE SYSTEM								BY DATABASE							
NUMBER	CLASS	20/20/20/20/20	EXP SMOOTHED	30/20/30/10/10	EXP SMOOTHED	50/30/00/10/10	EXP SMOOTHED	NUMBER	CLASS	20/20/20/20/20	EXP SMOOTHED	30/20/30/10/10	EXP SMOOTHED	50/30/00/10/10	EXP SMOOTHED
1	5	4	4.00	4	4.00	4	4.00	6	1	4	4.00	4	4.00	4	4.00
2	2	4	4.00	4	4.00	4	4.00	15	1	4	4.00	4	4.00	4	4.00
3	4	4	4.00	4	4.00	4	4.00	20	1	3	3.85	2	3.70	2	3.70
4	2	2	3.70	2	3.70	2	3.70	28	1	3	3.72	3	3.60	3	3.60
5	3	4	3.75	4	3.75	4	3.75	35	1	3	3.61	3	3.51	1	3.21
6	1	4	3.78	4	3.78	4	3.78	41	1	0	3.07	0	2.98	0	2.72
7	5	1	3.37	0	3.22	0	3.22	43	1	0	2.61	0	2.53	0	2.32
8	5	0	2.86	0	2.73	0	2.73	44	1	1	2.37	1	2.30	2	2.27
9	2	0	2.43	0	2.32	0	2.32	48	1	0	2.01	0	1.96	1	2.08
10	5	0	2.07	0	1.97	0	1.97	56	1	0	1.71	0	1.66	0	1.77
11	5	0	1.76	0	1.68	0	1.68	58	1	0	1.46	0	1.41	0	1.50
12	2	0	1.49	0	1.43	1	1.58	59	1	1	1.39	1	1.35	1	1.43
13	5	0	1.27	0	1.21	1	1.49	61	1	1	1.33	1	1.30	2	1.51
14	4	0	1.08	0	1.03	0	1.27	62	1	1	1.28	1	1.25	1	1.44
15	1	4	1.52	4	1.48	4	1.68	69	1	0	1.09	0	1.07	0	1.22
16	2	0	1.29	0	1.25	0	1.43	76	1	0	0.92	0	0.91	0	1.04
17	2	0	1.10	0	1.07	0	1.21	79	1	0	0.79	0	0.77	0	0.88
18	2	0	0.93	0	0.91	1	1.18	86	1	0	0.67	0	0.65	0	0.75
19	3	0	0.79	0	0.77	0	1.00	89	1	0	0.57	0	0.56	0	0.64
20	1	3	1.12	2	0.96	2	1.15	94	1	2	0.78	2	0.77	3	0.99
21	5	1	1.10	1	0.96	0	0.98	96	1	0	0.67	0	0.66	0	0.84
22	5	0	0.94	0	0.82	0	0.83	101	1	0	0.57	0	0.56	0	0.72
23	5	1	0.95	1	0.84	0	0.71	103	1	0	0.48	1	0.62	1	0.76
24	3	0	0.81	0	0.72	0	0.60	110	1	0	0.41	0	0.53	1	0.80
25	2	0	0.69	0	0.61	1	0.66	111	1	0	0.35	0	0.45	1	0.83
26	2	0	0.58	0	0.52	0	0.56	112	1	0	0.30	0	0.38	0	0.70
27	5	0	0.49	0	0.44	0	0.48	2	2	4	4.00	4	4.00	4	4.00
28	1	3	0.87	3	0.82	3	0.86	4	2	2	3.70	2	3.70	2	3.70
29	2	0	0.74	0	0.70	0	0.73	9	2	0	3.15	0	3.15	0	3.15
30	2	4	1.23	4	1.20	4	1.22	12	2	0	2.67	0	2.67	1	2.82
31	2	0	1.04	0	1.02	0	1.04	16	2	0	2.27	0	2.27	0	2.40
32	5	0	0.89	0	0.86	0	0.88	17	2	0	1.93	0	1.93	0	2.04
33	5	0	0.75	0	0.73	0	0.75	18	2	0	1.64	0	1.64	1	1.88
34	4	0	0.64	0	0.62	0	0.64	25	2	0	1.40	0	1.40	1	1.75
35	1	3	1.00	3	0.98	1	0.69	26	2	0	1.19	0	1.19	0	1.49
36	5	0	0.85	0	0.83	0	0.59	29	2	0	1.01	0	1.01	0	1.27
37	5	0	0.72	0	0.71	0	0.50	30	2	4	1.46	4	1.46	4	1.68
38	5	0	0.61	0	0.60	1	0.57	31	2	0	1.24	0	1.24	0	1.42
39	4	0	0.52	0	0.51	1	0.64	40	2	3	1.50	3	1.50	3	1.66
40	2	3	0.89	3	0.89	3	0.99	45	2	3	1.73	2	1.58	3	1.86
41	1	0	0.76	0	0.75	0	0.84	52	2	2	1.77	2	1.64	1	1.73
42	4	0	0.64	0	0.64	1	0.87	53	2	2	1.80	2	1.69	4	2.07
43	1	0	0.55	0	0.54	0	0.74	60	2	1	1.68	1	1.59	0	1.76
44	1	1	0.62	1	0.61	2	0.93	63	2	1	1.58	1	1.50	0	1.50
45	2	3	0.97	2	0.82	3	1.24	70	2	1	1.49	1	1.43	0	1.27
46	5	0	0.83	0	0.70	0	1.05	75	2	2	1.57	1	1.36	2	1.38
47	4	0	0.70	0	0.59	0	0.89	78	2	1	1.48	1	1.31	1	1.32
48	1	0	0.60	0	0.50	1	0.91	80	2	2	1.56	2	1.41	2	1.43
49	5	0	0.51	0	0.43	0	0.77	82	2	1	1.48	1	1.35	0	1.21
50	5	0	0.43	0	0.36	1	0.81	84	2	3	1.71	2	1.45	3	1.48
51	5	0	0.37	0	0.31	1	0.84	87	2	1	1.60	1	1.38	0	1.26
52	2	2	0.61	2	0.56	1	0.86	88	2	1	1.51	1	1.32	0	1.07
53	2	2	0.82	2	0.78	4	1.33	91	2	1	1.43	1	1.27	0	0.91
54	5	0	0.70	0	0.66	0	1.13	93	2	0	1.22	0	1.08	0	0.77
55	4	0	0.59	0	0.56	1	1.11	107	2	1	1.19	1	1.07	0	0.66
56	1	0	0.50	0	0.48	0	0.95	113	2	2	1.31	2	1.21	1	0.71
57	5	0	0.43	0	0.41	0	0.80	5	3	4	4.00	4	4.00	4	4.00
58	1	0	0.36	0	0.35	0	0.68	19	3	0	3.40	0	3.40	0	3.40
59	1	1	0.46	1	0.44	1	0.73	24	3	0	2.89	0	2.89	0	2.89
60	2	1	0.54	1	0.53	0	0.62	68	3	0	2.46	0	2.46	0	2.46
61	1	1	0.61	1	0.60	2	0.83	81	3	0	2.09	0	2.09	0	2.09
62	1	1	0.67	1	0.66	1	0.85	3	4	4	4.00	4	4.00	4	4.00
63	2	1	0.72	1	0.71	0	0.73	14	4	0	3.40	0	3.40	0	3.40

SAMPLE 2

WHOLE SYSTEM								BY CLASS							
NUMBER	CLASS	20/20/20/20/20	EXP SMOOTHED	30/20/30/10/10	EXP SMOOTHED	50/30/00/10/10	EXP SMOOTHED	NUMBER	CLASS	20/20/20/20/20	EXP SMOOTHED	30/20/30/10/10	EXP SMOOTHED	50/30/00/10/10	EXP SMOOTHED
1	5	4	4.00	4	4.00	4	4.00	6	1	4	4.00	4	4.00	4	4.00
2	2	4	4.00	4	4.00	4	4.00	15	1	0	3.40	0	3.40	0	3.40
3	4	4	4.00	4	4.00	4	4.00	20	1	0	2.89	0	2.89	1	3.04
4	2	1	3.55	1	3.55	1	3.55	28	1	3	2.91	2	2.76	4	3.18
5	3	4	3.62	4	3.62	4	3.62	35	1	0	2.47	0	2.34	0	2.71
6	1	4	3.67	4	3.67	4	3.67	41	1	0	2.10	0	1.99	1	2.45
7	5	0	3.12	0	3.12	0	3.12	43	1	1	1.93	1	1.84	1	2.23
8	5	2	2.96	0	2.66	0	2.66	44	1	0	1.64	0	1.57	1	2.05
9	2	4	3.11	4	2.86	4	2.86	48	1	0	1.40	0	1.33	2	2.04
10	5	2	2.95	2	2.73	2	2.73	56	1	1	1.34	0	1.13	0	1.73
11	5	0	2.50	0	2.32	1	2.47	58	1	0	1.14	0	0.96	0	1.47
12	2	1	2.28	1	2.12	2	2.40	59	1	1	1.12	1	0.97	1	1.40
13	5	1	2.09	1	1.95	2	2.34	61	1	0	0.95	0	0.82	1	1.34
14	4	0	1.77	0	1.66	0	1.99	62	1	0	0.81	0	0.70	0	1.14
15	1	3	1.96	2	1.71	4	2.29	69	1	0	0.69	0	0.59	0	0.97
16	2	0	1.66	0	1.45	0	1.95	76	1	0	0.58	0	0.51	0	0.82
17	2	1	1.56	1	1.39	2	1.95	79	1	0	0.50	0	0.43	0	0.70
18	2	1	1.48	1	1.33	1	1.81	86	1	0	0.42	0	0.36	0	0.60
19	3	2	1.56	2	1.43	0	1.54	89	1	0	0.36	0	0.31	0	0.51
20	1	0	1.32	0	1.21	0	1.31	94	1	1	0.45	1	0.41	1	0.58
21	5	1	1.28	1	1.18	2	1.41	96	1	0	0.39	0	0.35	1	0.64
22	5	2	1.38	2	1.31	3	1.65	101	1	1	0.48	1	0.45	2	0.85
23	5	1	1.33	1	1.26	1	1.55	103	1	0	0.41	0	0.38	1	0.87
24	3	0	1.13	0	1.07	1	1.47	110	1	0	0.35	0	0.32	1	0.89
25	2	0	0.96	0	0.91	1	1.40	111	1	1	0.44	1	0.43	2	1.06
26	2	0	0.81	0	0.77	0	1.19	112	1	0	0.38	0	0.36	0	0.90
27	5	0	0.69	0	0.66	0	1.01	2	2	4	4.00	4	4.00	4	4.00
28	1	1	0.74	1	0.71	0	0.86	4	2	1	3.55	1	3.55	1	3.55
29	2	1	0.78	1	0.75	1	0.88	9	2	4	3.62	4	3.62	4	3.62
30	2	0	0.66	0	0.64	0	0.75	12	2	2	3.37	2	3.37	2	3.37
31	2	1	0.71	1	0.69	0	0.64	16	2	1	3.02	1	3.02	2	3.17
32	5	0	0.61	0	0.59	0	0.54	17	2	1	2.72	1	2.72	2	2.99
33	5	0	0.51	0	0.50	0	0.46	18	2	1	2.46	1	2.46	2	2.84
34	4	0	0.44	0	0.43	0	0.39	25	2	1	2.24	1	2.24	1	2.57
35	1	0	0.37	0	0.36	1	0.48	26	2	1	2.05	1	2.05	2	2.48
36	5	2	0.62	2	0.61	4	1.01	29	2	2	2.05	2	2.05	3	2.56
37	5	2	0.82	2	0.82	4	1.46	30	2	1	1.89	1	1.89	1	2.33
38	5	2	1.00	2	0.99	4	1.84	31	2	1	1.76	1	1.76	0	1.98
39	4	1	1.00	1	1.00	0	1.56	40	2	1	1.64	1	1.64	0	1.68
40	2	0	0.85	0	0.85	0	1.33	45	2	2	1.70	2	1.70	4	2.03
41	1	0	0.72	0	0.72	0	1.13	52	2	2	1.74	2	1.74	4	2.32
42	4	0	0.61	0	0.61	2	1.26	53	2	1	1.63	1	1.63	0	1.98
43	1	1	0.67	0	0.52	0	1.07	60	2	1	1.54	1	1.54	1	1.83
44	1	0	0.57	0	0.44	0	0.91	63	2	1	1.46	1	1.46	1	1.70
45	2	1	0.64	1	0.53	1	0.92	70	2	0	1.24	0	1.24	0	1.45
46	5	1	0.69	1	0.60	1	0.94	75	2	3	1.50	3	1.50	3	1.68
47	4	0	0.59	0	0.51	0	0.80	78	2	3	1.73	3	1.73	3	1.88
48	1	0	0.50	0	0.43	0	0.68	80	2	0	1.47	0	1.47	1	1.75
49	5	0	0.42	0	0.37	0	0.57	82	2	0	1.25	0	1.25	0	1.49
50	5	0	0.36	0	0.31	0	0.49	84	2	1	1.21	1	1.21	1	1.41
51	5	0	0.31	0	0.26	0	0.42	87	2	0	1.03	0	1.03	0	1.20
52	2	1	0.41	2	0.52	0	0.35	88	2	0	0.87	0	0.87	0	1.02
53	2	0	0.35	0	0.45	0	0.30	91	2	1	0.89	1	0.89	1	1.02
54	5	3	0.75	3	0.83	3	0.70	93	2	0	0.76	0	0.76	0	0.86
55	4	3	1.08	3	1.15	3	1.05	107	2	2	0.95	2	0.95	2	1.04
56	1	0	0.92	0	0.98	0	0.89	113	2	1	0.95	1	0.95	0	0.88
57	5	1	0.93	1	0.98	0	0.76	5	3	4	4.00	4	4.00	4	4.00
58	1	3	1.24	3	1.29	0	0.64	19	3	0	3.40	0	3.40	0	3.40
59	1	0	1.06	0	1.09	0	0.55	24	3	0	2.89	0	2.89	0	2.89
60	2	0	0.90	0	0.93	0	0.47	68	3	0	2.46	0	2.46	0	2.46
61	1	0	0.76	0	0.79	0	0.40	81	3	0	2.09	0	2.09	0	2.09
62	1	0	0.65	0	0.67	0	0.34	3	4	4	4.00	4	4.00	4	4.00
63	2	0	0.55	0	0.57	0	0.29	14	4	2	3.70	2	3.70	4	4.00

SAMPLE 1																		
WHOLE SYSTEM									BY DATABASE									
64	5	0	0.61	0	0.60	0	0.62		34	4	0	2.89	0	2.89	0	2.89		
65	4	1	0.67	1	0.66	2	0.82		39	4	0	2.46	0	2.46	1	2.61		
66	5	0	0.57	0	0.56	0	0.70		42	4	0	2.09	0	2.09	1	2.37		
67	5	1	0.63	1	0.63	1	0.75		47	4	0	1.77	0	1.77	0	2.01		
68	3	0	0.54	0	0.53	0	0.63		55	4	0	1.51	0	1.51	1	1.86		
69	1	0	0.46	0	0.45	0	0.54		65	4	1	1.43	1	1.43	2	1.88		
70	2	1	0.54	1	0.54	0	0.46		73	4	1	1.37	1	1.37	2	1.90		
71	5	2	0.76	3	0.91	0	0.39		90	4	0	1.16	0	1.16	0	1.61		
72	5	0	0.64	0	0.77	0	0.33		100	4	0	0.99	0	0.99	0	1.37		
73	4	1	0.70	1	0.80	2	0.58		108	4	0	0.84	0	0.84	0	1.17		
74	5	0	0.59	0	0.68	0	0.49		109	4	0	0.71	0	0.71	0	0.99		
75	2	2	0.80	1	0.73	2	0.72		1	5	4	4.00	4	4.00	4	4.00		
76	1	0	0.68	0	0.62	0	0.61		7	5	1	3.55	0	3.40	0	3.40		
77	5	1	0.73	3	0.98	0	0.52		8	5	0	3.02	0	2.89	0	2.89		
78	2	1	0.77	1	0.98	1	0.59		10	5	0	2.56	0	2.46	0	2.46		
79	1	0	0.66	0	0.83	0	0.50		11	5	0	2.18	0	2.09	0	2.09		
80	2	2	0.86	2	1.01	2	0.73		13	5	0	1.85	0	1.77	1	1.92		
81	3	0	0.73	0	0.86	0	0.62		21	5	1	1.73	1	1.66	0	1.64		
82	2	1	0.77	1	0.88	0	0.53		22	5	0	1.47	0	1.41	0	1.39		
83	5	0	0.65	0	0.75	0	0.45		23	5	1	1.40	1	1.35	0	1.18		
84	2	3	1.01	2	0.94	3	0.83		27	5	0	1.19	0	1.15	0	1.00		
85	5	0	0.85	0	0.79	0	0.71		32	5	0	1.01	0	0.97	0	0.85		
86	1	0	0.73	0	0.68	0	0.60		33	5	0	0.86	0	0.83	0	0.73		
87	2	1	0.77	1	0.72	0	0.51		36	5	0	0.73	0	0.70	0	0.62		
88	2	1	0.80	1	0.77	0	0.43		37	5	0	0.62	0	0.60	0	0.52		
89	1	0	0.68	0	0.65	0	0.37		38	5	0	0.53	0	0.51	1	0.60		
90	4	0	0.58	0	0.55	0	0.31		46	5	0	0.45	0	0.43	0	0.51		
91	2	1	0.64	1	0.62	0	0.27		49	5	0	0.38	0	0.37	0	0.43		
92	5	0	0.55	0	0.53	0	0.23		50	5	0	0.32	0	0.31	1	0.52		
93	2	0	0.46	0	0.45	0	0.19		51	5	0	0.27	0	0.27	1	0.59		
94	1	2	0.69	2	0.68	3	0.61		54	5	0	0.23	0	0.23	0	0.50		
95	5	0	0.59	0	0.58	0	0.52		57	5	0	0.20	0	0.19	0	0.43		
96	1	0	0.50	0	0.49	0	0.44		64	5	0	0.17	0	0.16	0	0.36		
97	5	0	0.43	0	0.42	0	0.38		66	5	0	0.14	0	0.14	0	0.31		
98	5	0	0.36	0	0.36	0	0.32		67	5	1	0.27	1	0.27	1	0.41		
99	5	0	0.31	0	0.30	0	0.27		71	5	2	0.53	3	0.68	0	0.35		
100	4	0	0.26	0	0.26	0	0.23		72	5	0	0.45	0	0.58	0	0.30		
101	1	0	0.22	0	0.22	0	0.20		74	5	0	0.38	0	0.49	0	0.25		
102	5	0	0.19	0	0.19	0	0.17		77	5	1	0.48	3	0.87	0	0.21		
103	1	0	0.16	1	0.31	1	0.29		83	5	0	0.40	0	0.74	0	0.18		
104	5	0	0.14	0	0.26	0	0.25		85	5	0	0.34	0	0.63	0	0.16		
105	5	0	0.12	0	0.22	1	0.36		92	5	0	0.29	0	0.53	0	0.13		
106	5	0	0.10	0	0.19	0	0.31		95	5	0	0.25	0	0.45	0	0.11		
107	2	1	0.23	1	0.31	0	0.26		97	5	0	0.21	0	0.38	0	0.10		
108	4	0	0.20	0	0.26	0	0.22		98	5	0	0.18	0	0.33	0	0.08		
109	4	0	0.17	0	0.22	0	0.19		99	5	0	0.15	0	0.28	0	0.07		
110	1	0	0.14	0	0.19	1	0.31		102	5	0	0.13	0	0.24	0	0.06		
111	1	0	0.12	0	0.16	1	0.41		104	5	0	0.11	0	0.20	0	0.05		
112	1	0	0.10	0	0.14	0	0.35		105	5	0	0.09	0	0.17	1	0.19		
113	2	2	0.4	2	0.4	1	0.4		106	5	0	0.1	0	0.1	0	0.2		

SAMPLE 2																		
WHOLE SYSTEM									BY CLASS									
64	5	1	0.62	1	0.64	1	0.39		34	4	0	3.15	0	3.15	0	3.40		
65	4	0	0.53	0	0.54	1	0.48		39	4	0	2.67	0	2.67	0	2.89		
66	5	0	0.45	0	0.46	1	0.56		42	4	0	2.27	0	2.27	0	2.46		
67	5	0	0.38	0	0.39	1	0.63		47	4	1	2.08	1	2.08	0	2.09		
68	3	0	0.32	0	0.33	0	0.53		55	4	0	1.77	0	1.77	0	1.77		
69	1	0	0.27	0	0.28	0	0.45		65	4	0	1.50	0	1.50	0	1.51		
70	2	0	0.23	0	0.24	0	0.39		73	4	1	1.43	1	1.43	1	1.43		
71	5	3	0.65	3	0.65	0	0.33		90	4	0	1.21	0	1.21	0	1.22		
72	5	1	0.70	1	0.71	1	0.43		100	4	0	1.03	0	1.03	0	1.03		
73	4	0	0.60	0	0.60	0	0.36		108	4	0	0.88	0	0.88	0	0.88		
74	5	1	0.66	1	0.66	1	0.46		109	4	0	0.75	0	0.75	1	0.90		
75	2	0	0.56	0	0.56	0	0.39		1	5	4	4.00	4	4.00	4	4.00		
76	1	0	0.47	0	0.48	0	0.33		7	5	2	3.70	0	3.40	0	3.40		
77	5	0	0.40	0	0.41	0	0.28		8	5	0	3.15	0	2.89	0	2.89		
78	2	0	0.34	0	0.34	0	0.24		10	5	0	2.67	0	2.46	0	2.46		
79	1	0	0.29	0	0.29	0	0.20		11	5	2	2.57	2	2.39	0	2.09		
80	2	3	0.70	3	0.70	0	0.17		13	5	0	2.19	0	2.03	1	1.92		
81	3	0	0.59	0	0.59	0	0.15		21	5	0	1.86	0	1.73	0	1.64		
82	2	0	0.50	0	0.50	0	0.13		22	5	0	1.58	0	1.47	0	1.39		
83	5	0	0.43	0	0.43	0	0.11		23	5	0	1.34	0	1.25	0	1.18		
84	2	0	0.36	0	0.36	0	0.09		27	5	0	1.14	0	1.06	0	1.00		
85	5	0	0.31	0	0.31	0	0.08		32	5	0	0.97	0	0.90	0	0.85		
86	1	0	0.26	0	0.26	0	0.07		33	5	0	0.82	0	0.77	0	0.73		
87	2	0	0.22	0	0.22	0	0.06		36	5	0	0.70	0	0.65	0	0.62		
88	2	1	0.34	1	0.34	0	0.05		37	5	0	0.60	0	0.55	0	0.52		
89	1	0	0.29	0	0.29	0	0.04		38	5	0	0.51	0	0.47	0	0.45		
90	4	0	0.25	0	0.25	0	0.03		46	5	1	0.58	2	0.70	0	0.38		
91	2	0	0.21	0	0.21	0	0.03		49	5	0	0.49	0	0.59	0	0.32		
92	5	0	0.18	0	0.18	0	0.02		50	5	3	0.87	3	0.96	0	0.27		
93	2	0	0.15	0	0.15	0	0.02		51	5	0	0.74	0	0.81	0	0.23		
94	1	1	0.28	1	0.28	1	0.17		54	5	0	0.63	0	0.69	0	0.20		
95	5	1	0.39	1	0.39	1	0.29		57	5	0	0.53	0	0.59	0	0.17		
96	1	0	0.33	0	0.33	1	0.40		64	5	0	0.45	0	0.50	1	0.29		
97	5	0	0.28	0	0.28	0	0.34		66	5	0	0.39	0	0.42	0	0.25		
98	5	0	0.24	0	0.24	0	0.29		67	5	3	0.78	3	0.81	0	0.21		
99	5	0	0.20	0	0.20	0	0.24		71	5	0	0.66	0	0.69	0	0.18		
100	4	0	0.17	0	0.17	1	0.36		72	5	3	1.01	3	1.04	0	0.15		
101	1	1	0.30	1	0.30	2	0.60		74	5	0	0.86	0	0.88	0	0.13		
102	5	0	0.25	0	0.25	0	0.51		77	5	0	0.73	0	0.75	0	0.11		
103	1	0	0.21	0	0.21	0	0.44		83	5	0	0.62	0	0.64	0	0.09		
104	5	0	0.18	0	0.18	1	0.52		85	5	0	0.53	0	0.54	0	0.08		
105	5	2	0.45	2	0.45	2	0.74		92	5	1	0.60	1	0.61	0	0.07		
106	5	1	0.54	1	0.54	0	0.63		95	5	0	0.51	0	0.52	0	0.06		
107	2	0	0.46	0	0.46	1	0.69		97	5	0	0.43	0	0.44	0	0.05		
108	4	0	0.39	0	0.39	0	0.58		98	5	0	0.37	0	0.37	0	0.04		
109	4	0	0.33	0	0.33	0	0.50		99	5	0	0.31	0	0.32	1	0.19		
110	1	1	0.43	1	0.43	2	0.72		102	5	0	0.27	0	0.27	0	0.16		
111	1	0	0.37	0	0.37	0	0.61		104	5	0	0.23	0	0.23	0	0.13		
112	1	0	0.31	0	0.31	1	0.67		105	5	0	0.19	0	0.20	0	0.11		
113	2	0	0.3	0	0.3	0	0.6		106	5	0	0.2	0	0.2	0	0.1		

Appendix M: Re Class 1 Charts

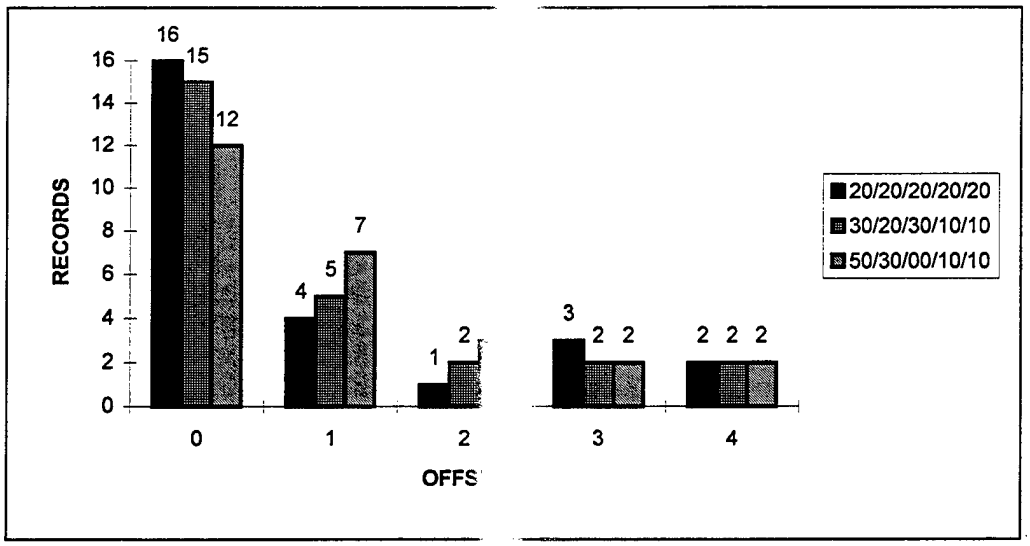

Figure 34. Histogram of mple 1 Results for Class 1

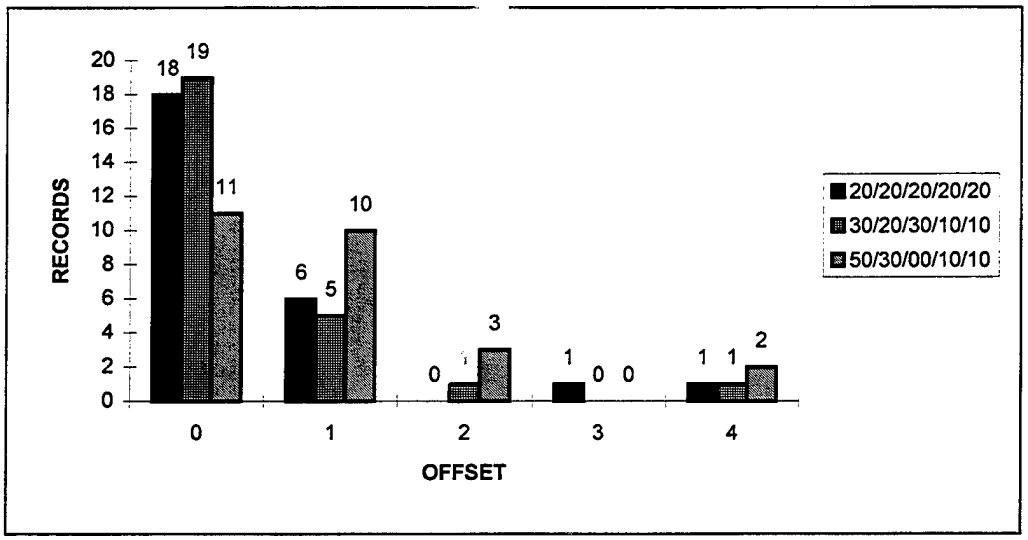

Figure 35. Histogram of Sample 2 Results for Class 1

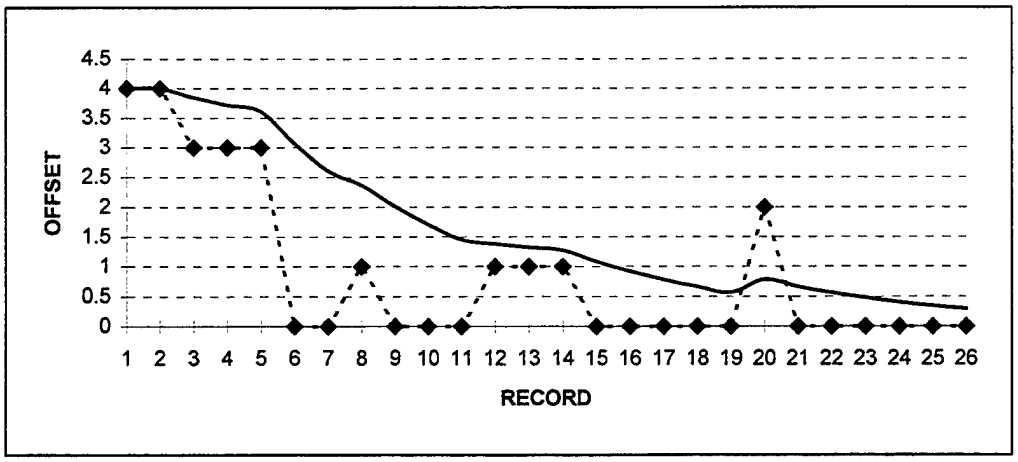

Figure 36. Time Series Results of Sample 1 for Class 1 with Weighting Scheme 20/20/20/20/20

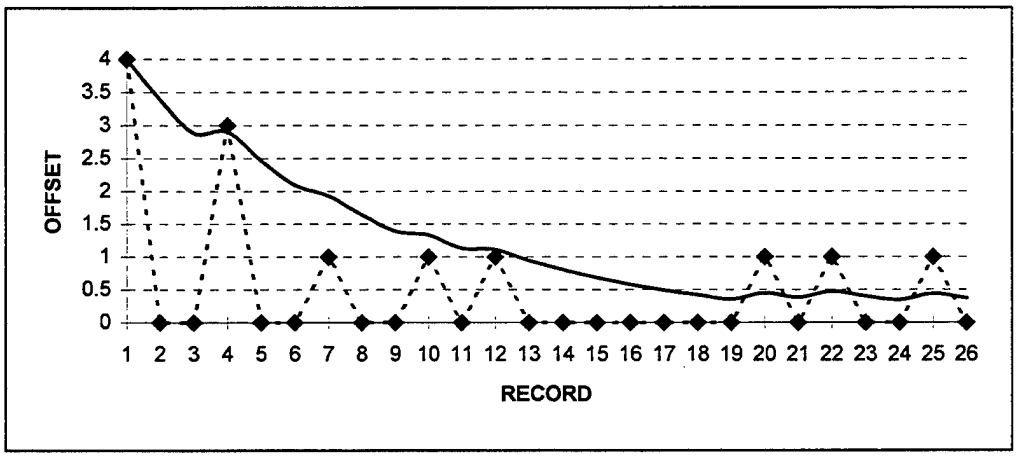

Figure 37. Time Series Results of Sample 2 for Class 1 with Weighting Scheme 20/20/20/20/20

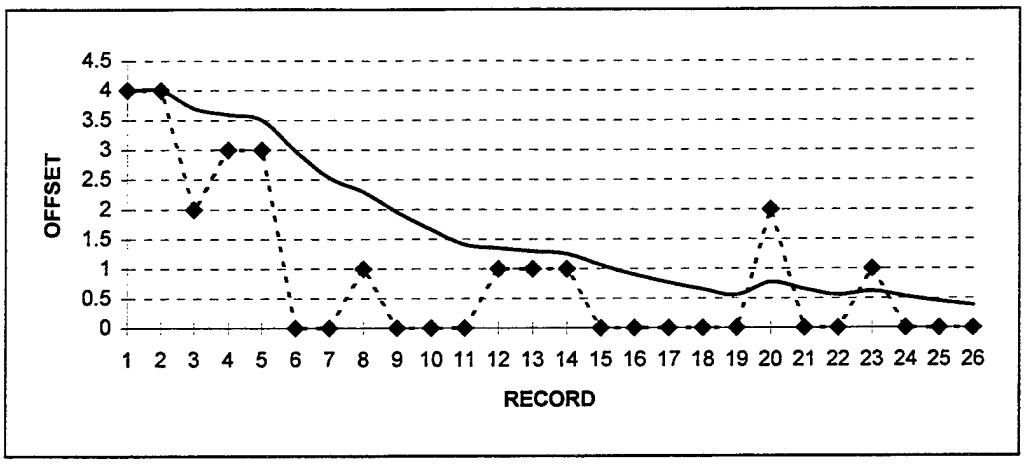

Figure 38. Time Series Results of Sample 1 for Class 1 with Weighting Scheme 30/20/30/10/10

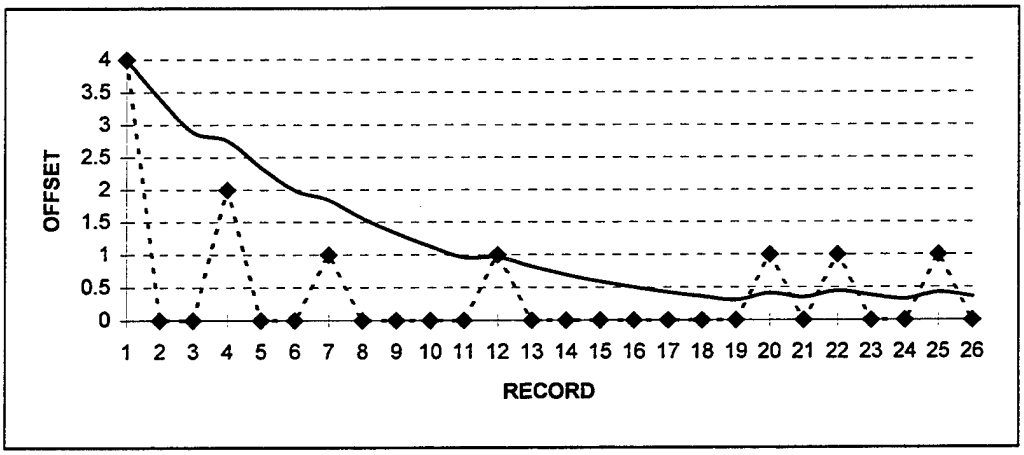

Figure 39. Time Series Results of Sample 2 for Class 1 with Weighting Scheme 30/20/30/10/10

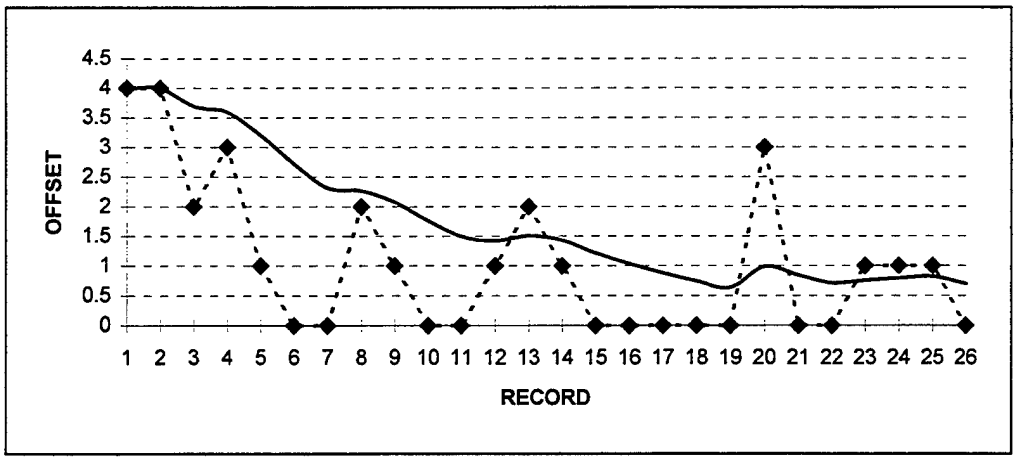

Figure 40. Time Series Results of Sample 1 for Class 1 with Weighting Scheme 50/30/00/10/10

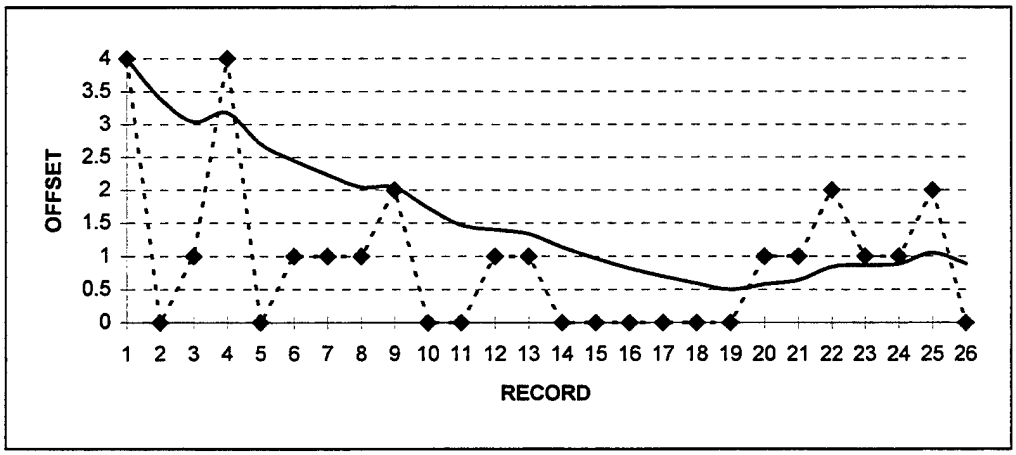

Figure 41. Time Series Results of Sample 2 for Class 1 with Weighting Scheme 50/30/00/10/10

Appendix N: Record Class 2 Charts

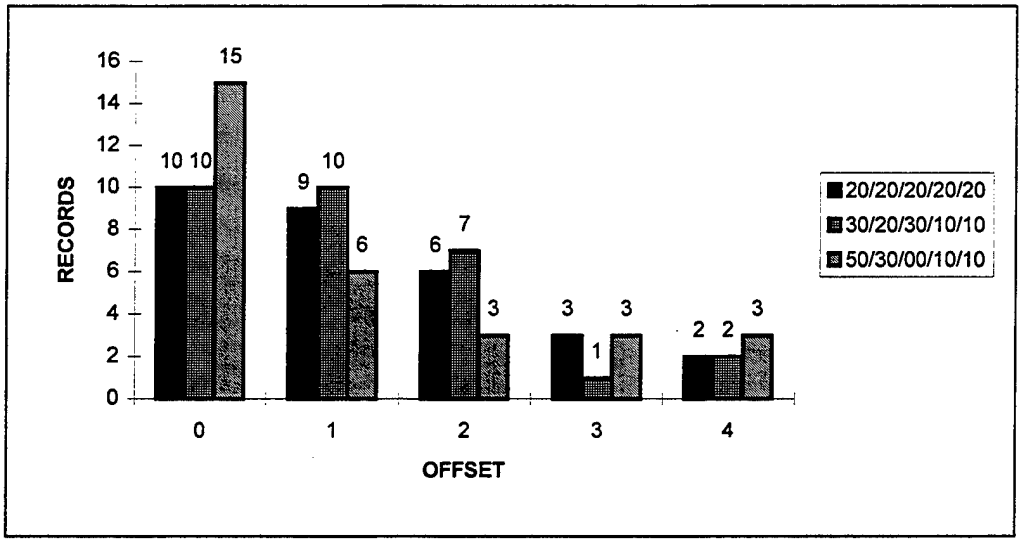

Figure 42. Histogram of Sample 1 Results for Class 2

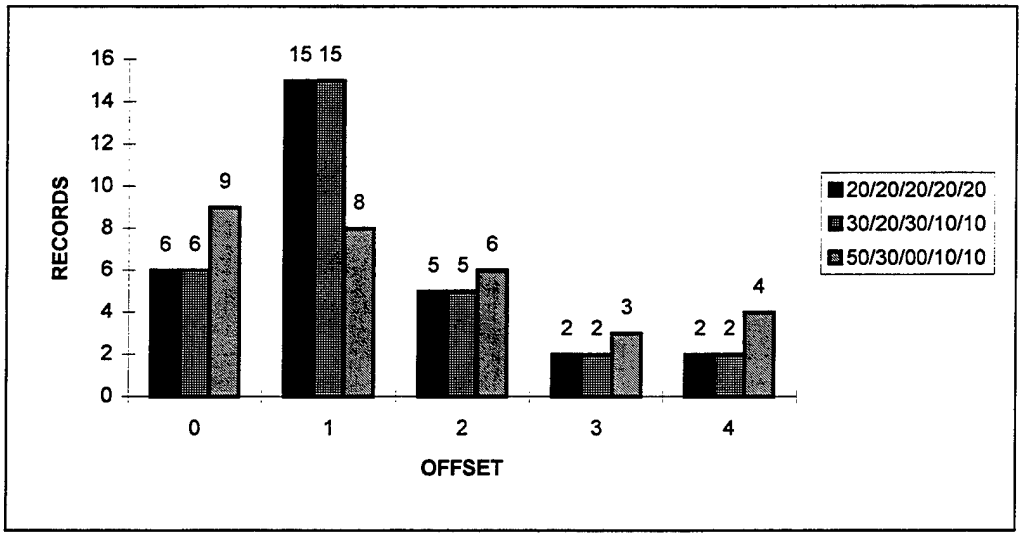

Figure 43. Histogram of Sample 2 Results for Class 2

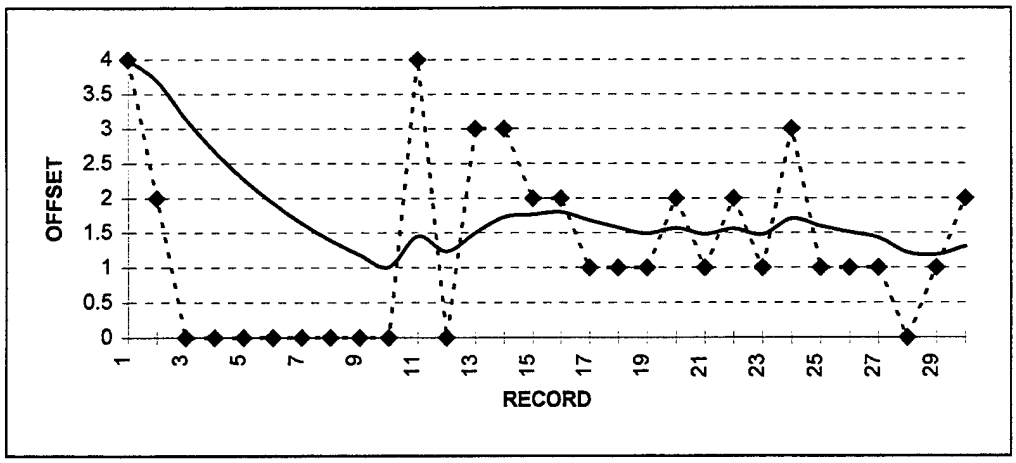

Figure 44. Time Series Results of Sample 1 for Class 2 with Weighting Scheme 20/20/20/20/20

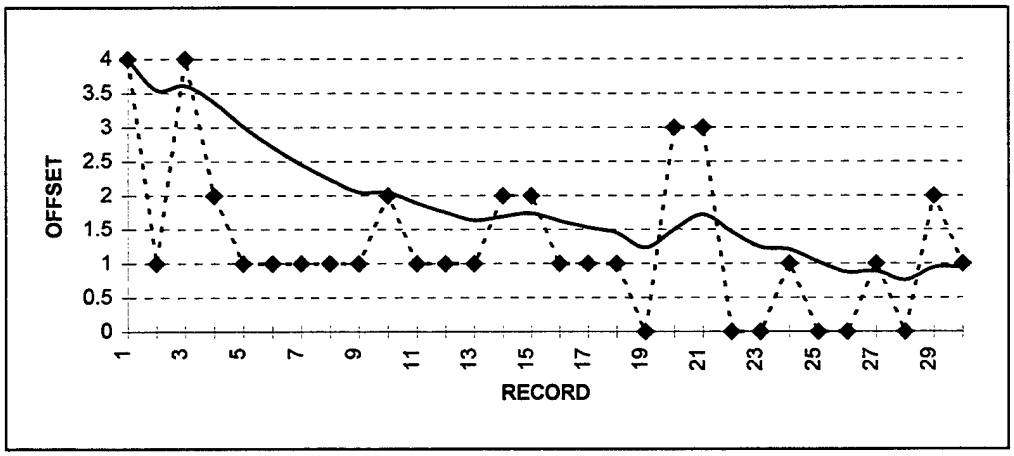

Figure 45. Time Series Results of Sample 2 for Class 2 with Weighting Scheme 20/20/20/20/20

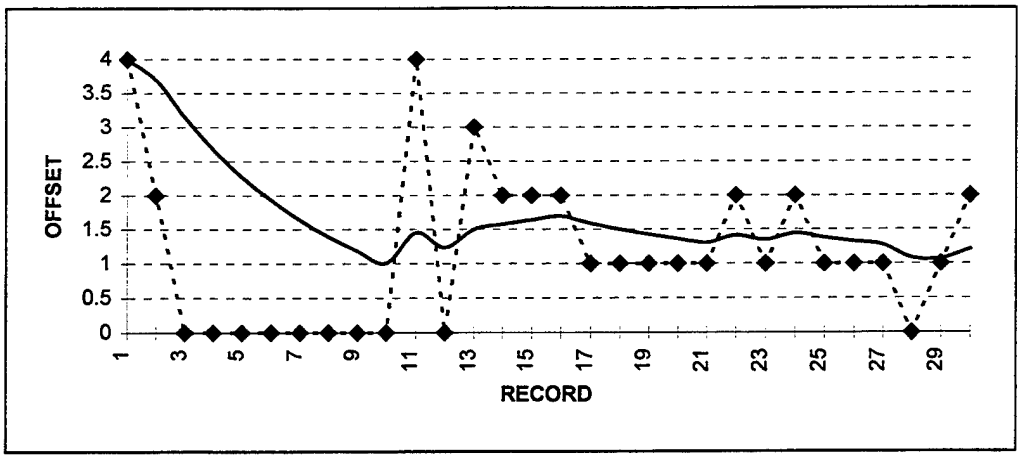

Figure 46. Time Series Results of Sample 1 for Class 2 with Weighting Scheme 30/20/30/10/10

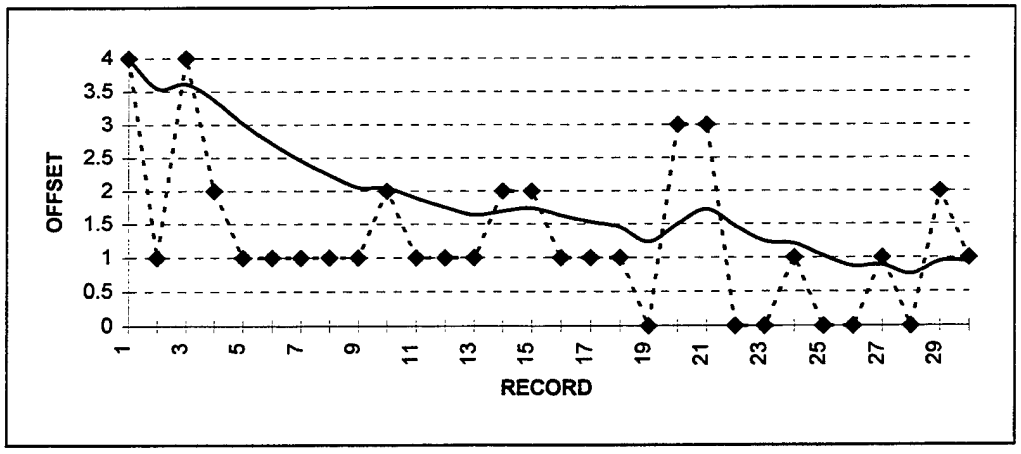

Figure 47. Time Series Results of Sample 2 for Class 2 with Weighting Scheme 30/20/30/10/10

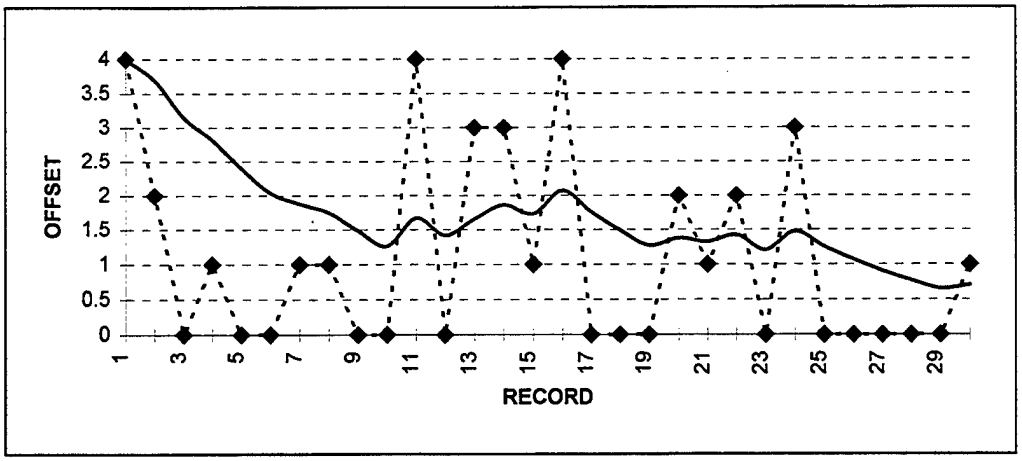

Figure 48. Time Series Results of Sample 1 for Class 2 with Weighting Scheme 50/30/00/10/10

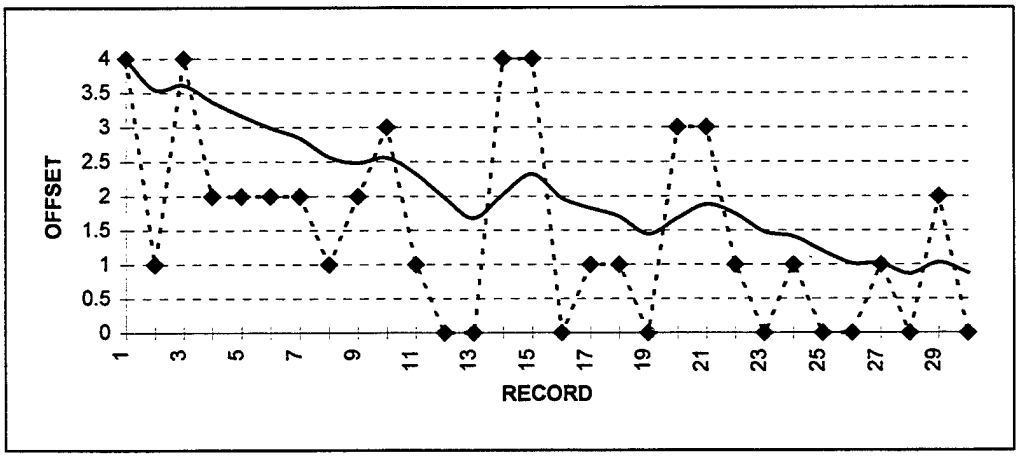

Figure 49. Time Series Results of Sample 2 for Class 2 with Weighting Scheme 50/30/00/10/10

Appendix O: Record Class 3 Charts

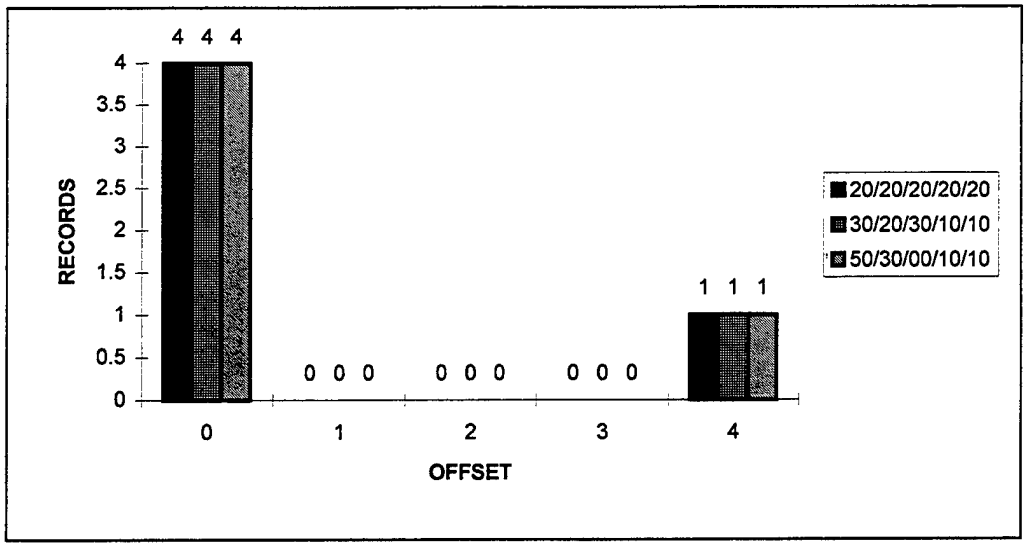

Figure 50. Histogram of Sample 1 Results for Class 3

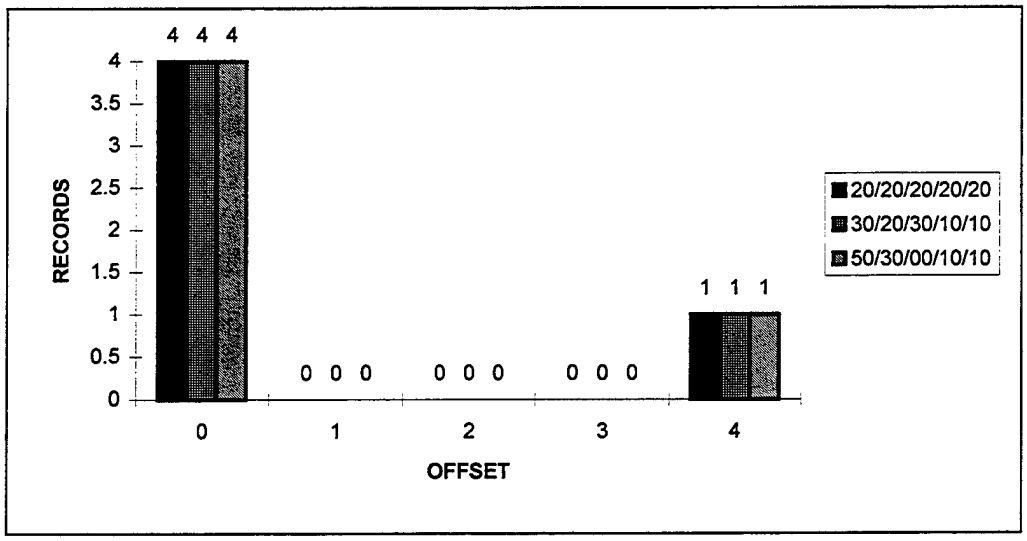

Figure 51. Histogram of Sample 2 Results for Class 3

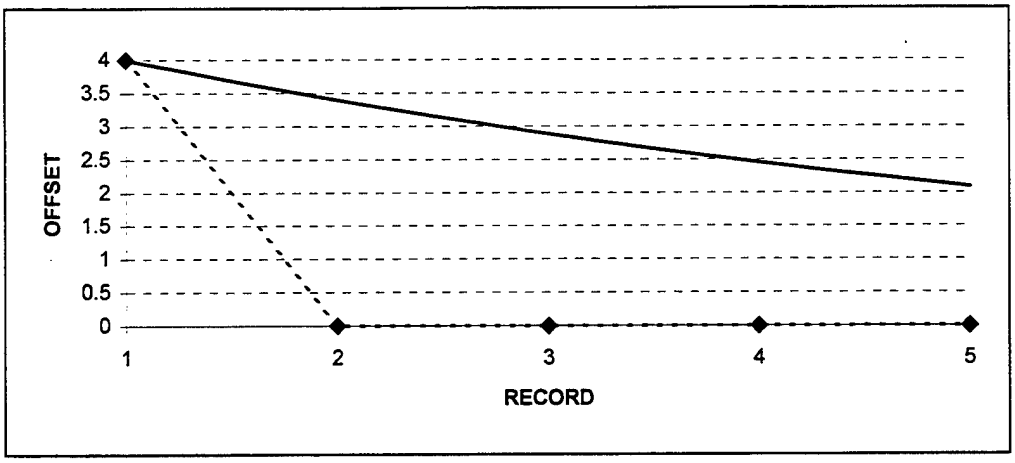

Figure 52. Time Series Results of Sample 1 for Class 3 with Weighting Scheme 20/20/20/20/20

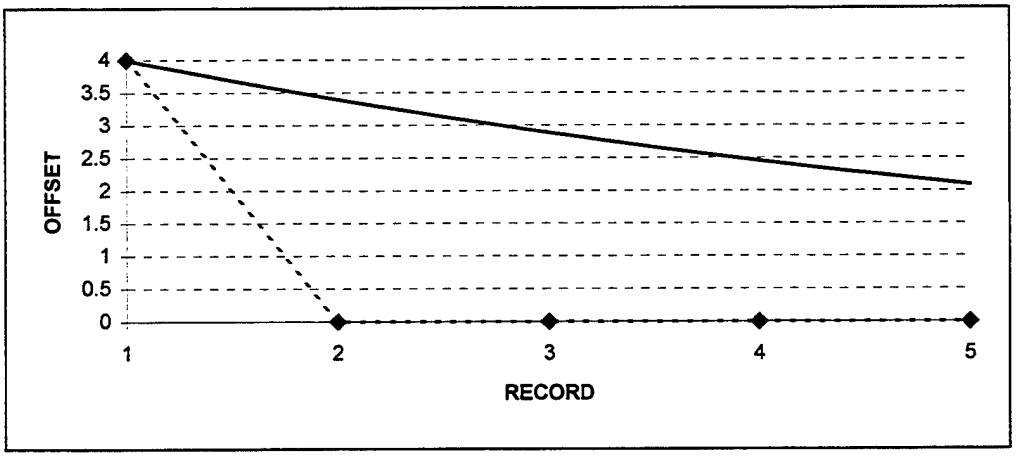

Figure 53. Time Series Results of Sample 2 for Class 3 with Weighting Scheme 20/20/20/20/20

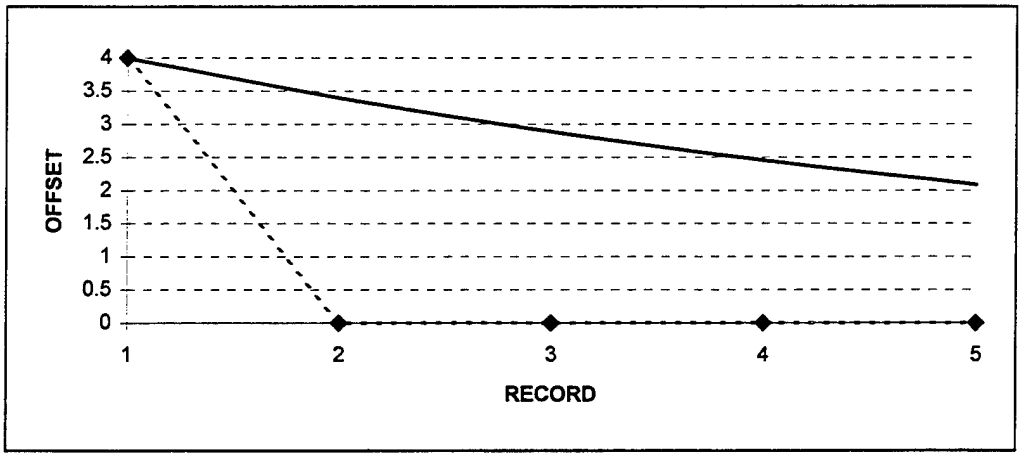

Figure 54. Time Series Results of Sample 1 for Class 3 with Weighting Scheme 30/20/30/10/10

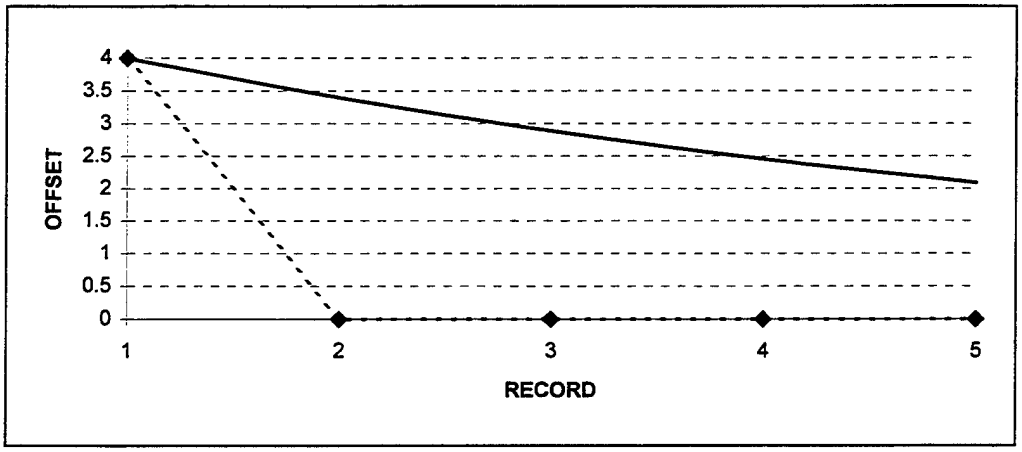

Figure 55. Time Series Results of Sample 2 for Class 3 with Weighting Scheme 30/20/30/10/10

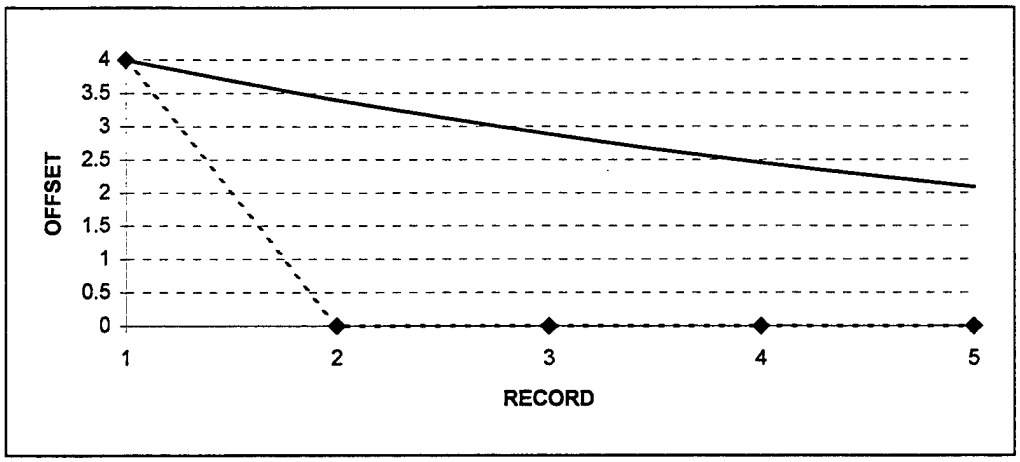

Figure 56. Time Series Results of Sample 1 for Class 3 with Weighting Scheme 50/30/00/10/10

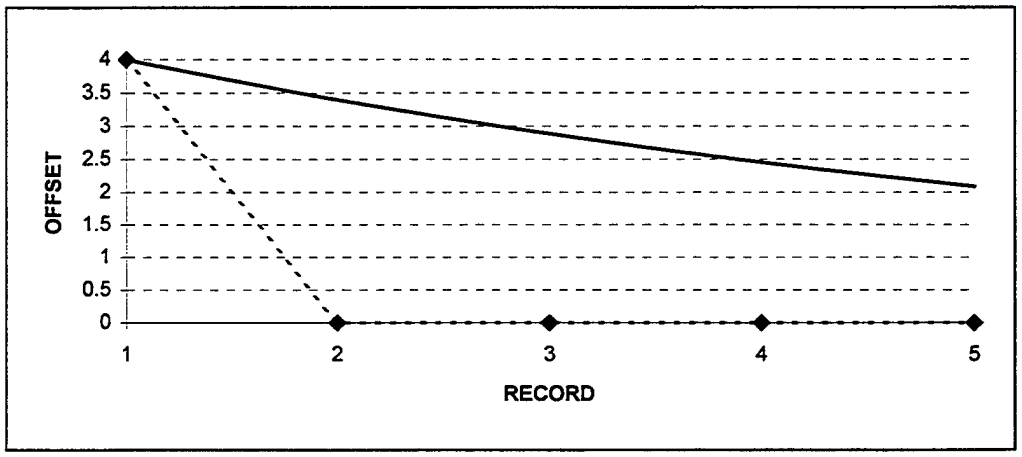

Figure 57. Time Series Results of Sample 2 for Class 3 with Weighting Scheme 50/30/00/10/10

Appendix P: Record Class 4 Charts

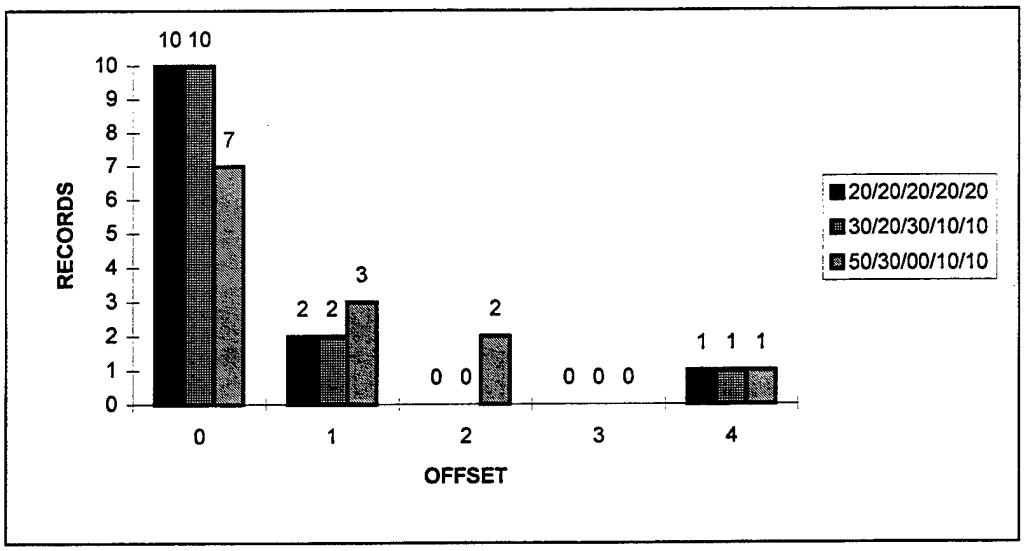

Figure 58. Histogram of Sample 1 Results for Class 4

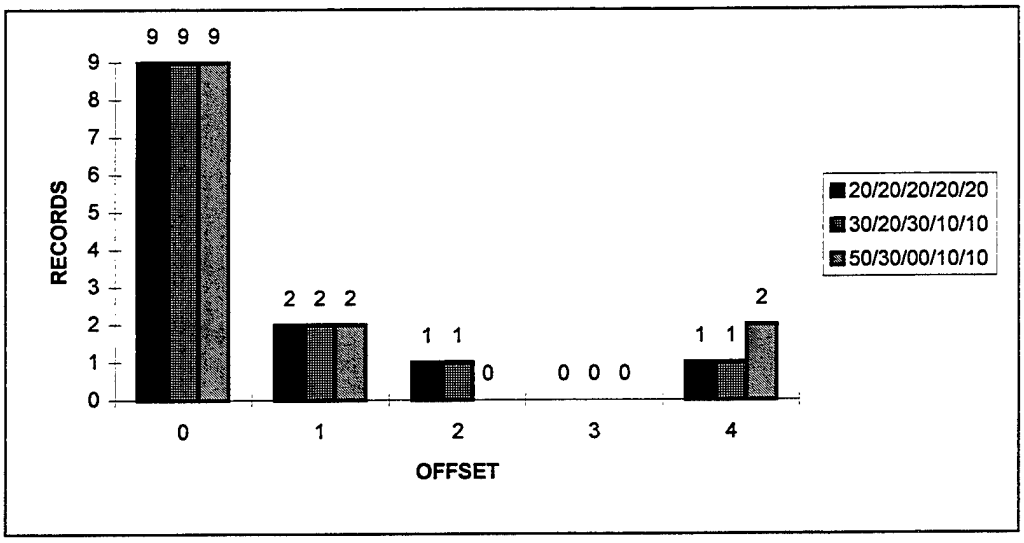

Figure 59. Histogram of Sample 2 Results for Class 4

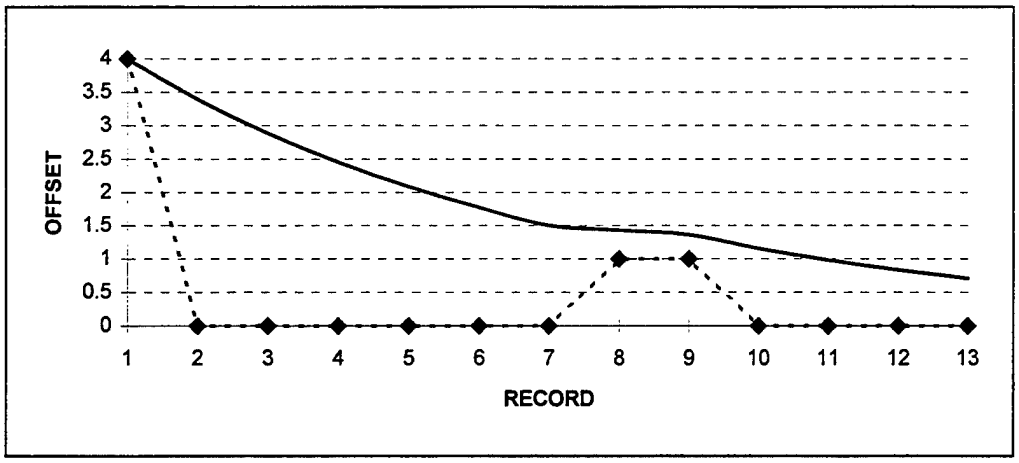

Figure 60. Time Series Results of Sample 1 for Class 4 with Weighting Scheme 20/20/20/20/20

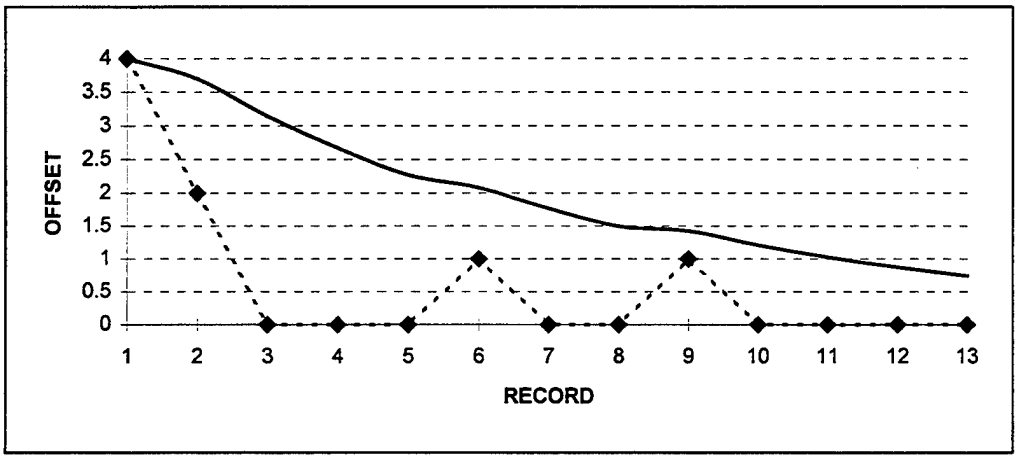

Figure 61. Time Series Results of Sample 2 for Class 4 with Weighting Scheme 20/20/20/20/20

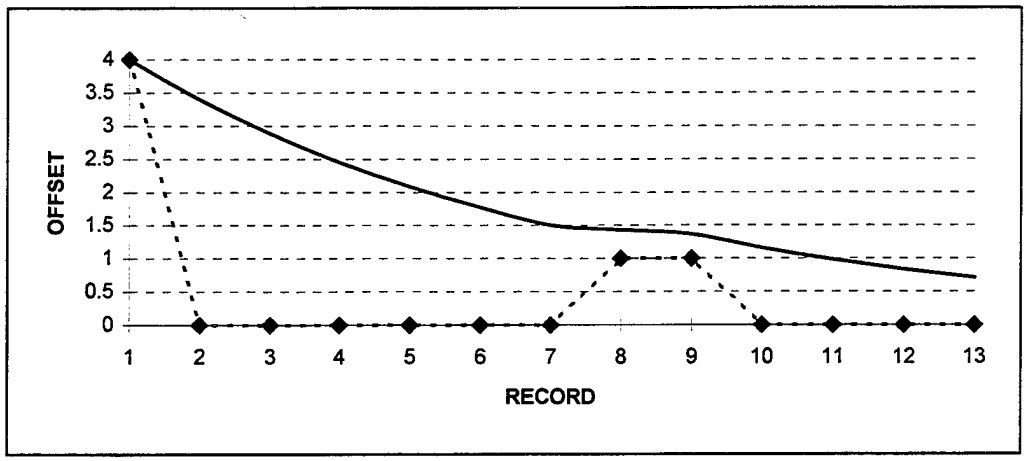

Figure 62. Time Series Results of Sample 1 for Class 4 with Weighting Scheme 30/20/30/10/10

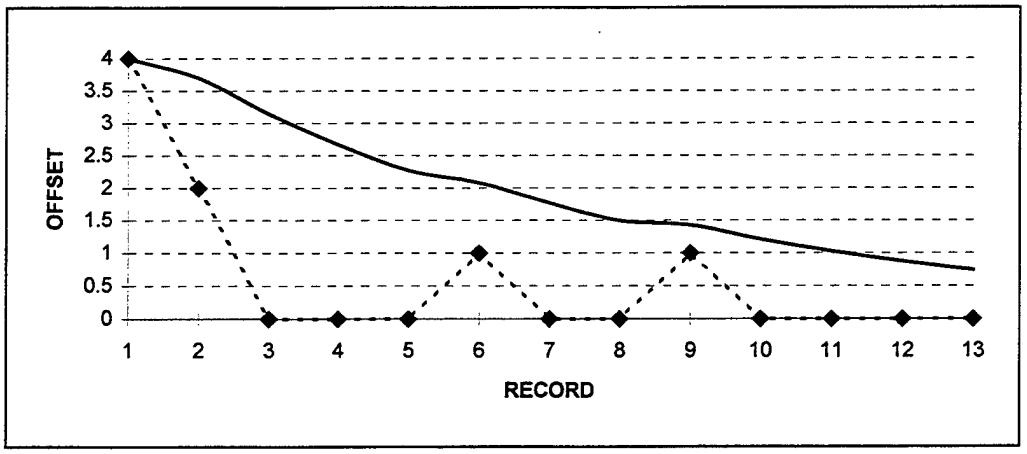

Figure 63. Time Series Results of Sample 2 for Class 4 with Weighting Scheme 30/20/30/10/10

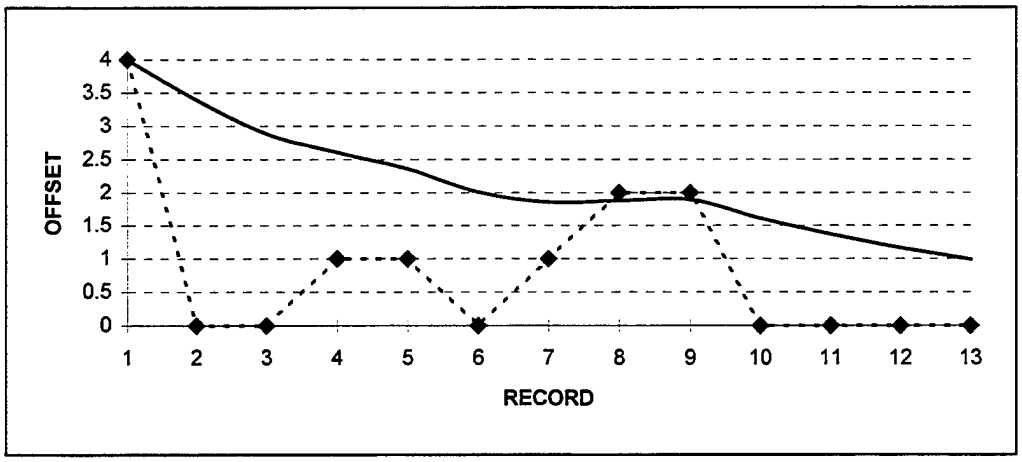

Figure 64. Time Series Results of Sample 1 for Class 4 with Weighting Scheme 50/30/00/10/10

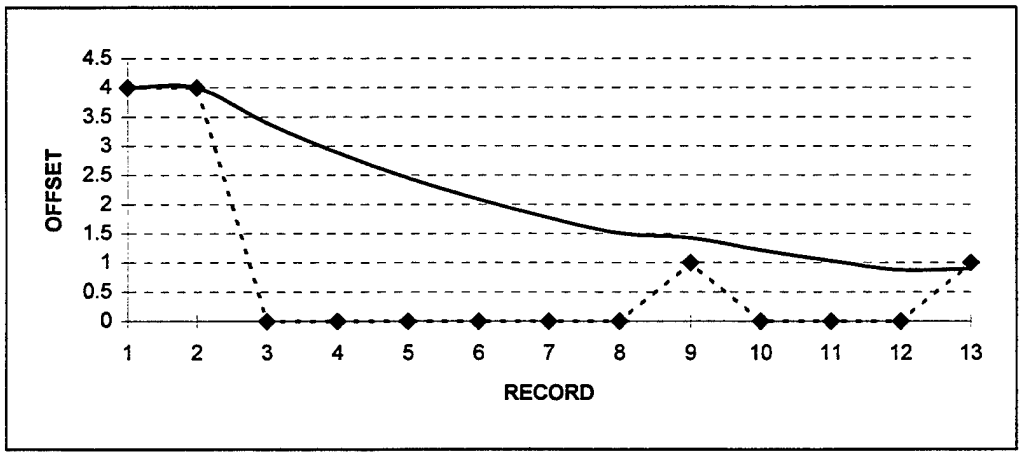

Figure 65. Time Series Results of Sample 2 for Class 4 with Weighting Scheme 50/30/00/10/10

Appendix Q: Record Class 5 Charts

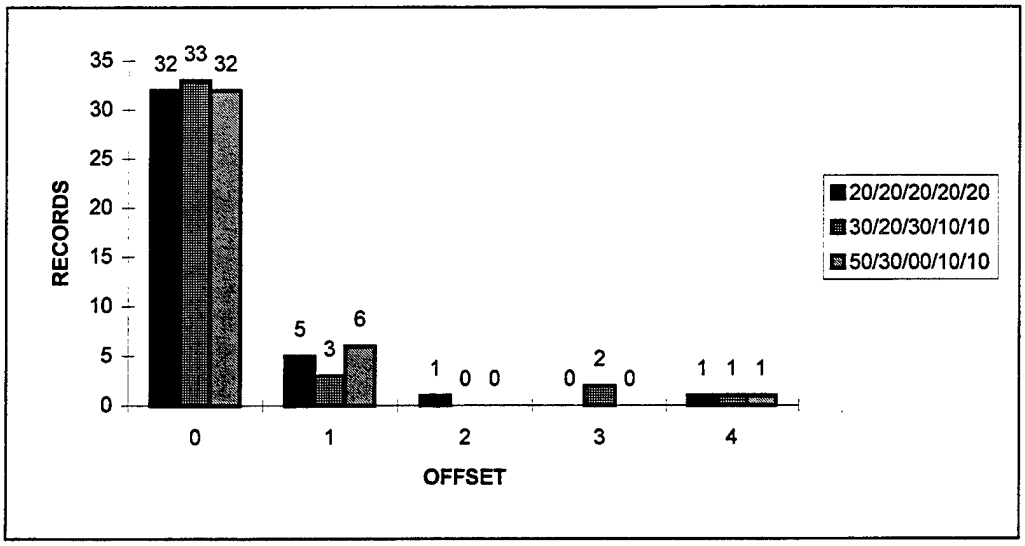

Figure 66. Histogram of Sample 1 Results for Class 5

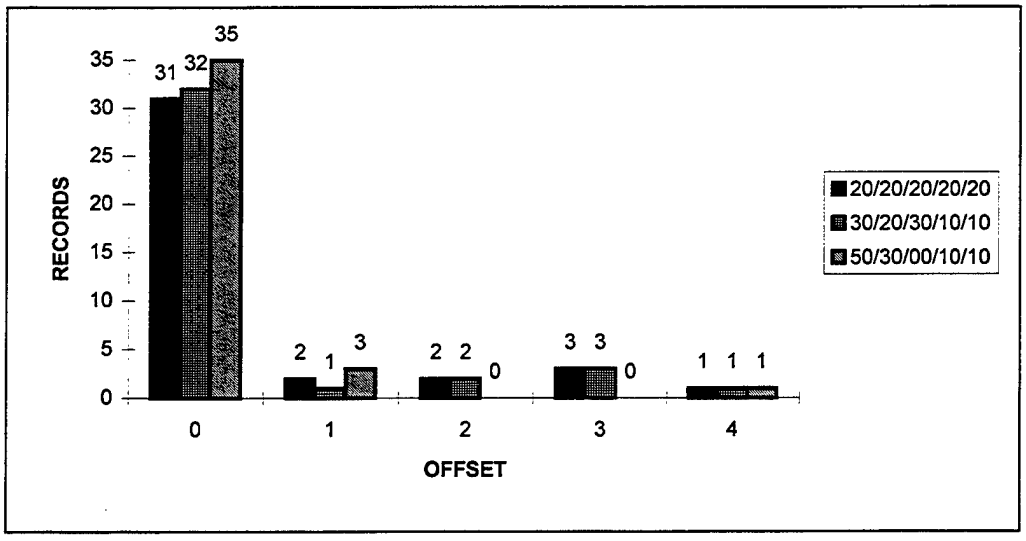

Figure 67. Histogram of Sample 2 Results for Class 5

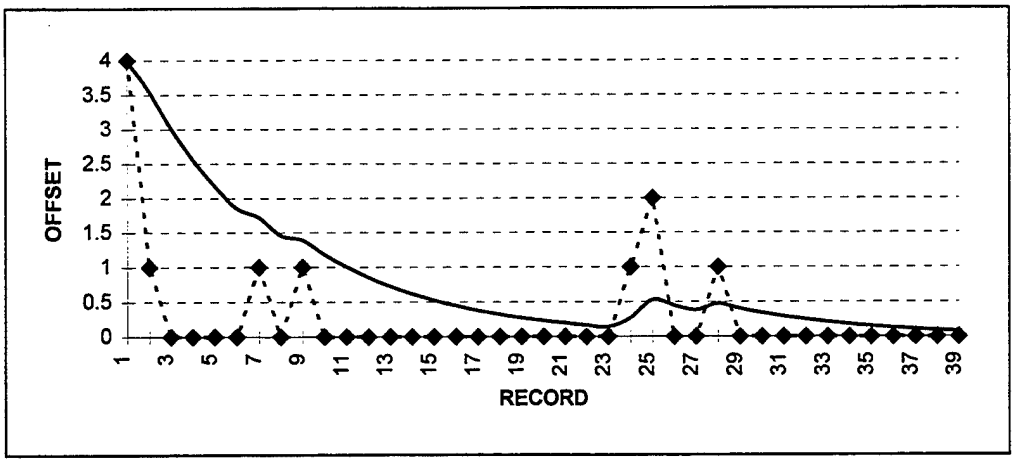

Figure 68. Time Series Results of Sample 1 for Class 5 with Weighting Scheme 20/20/20/20/20

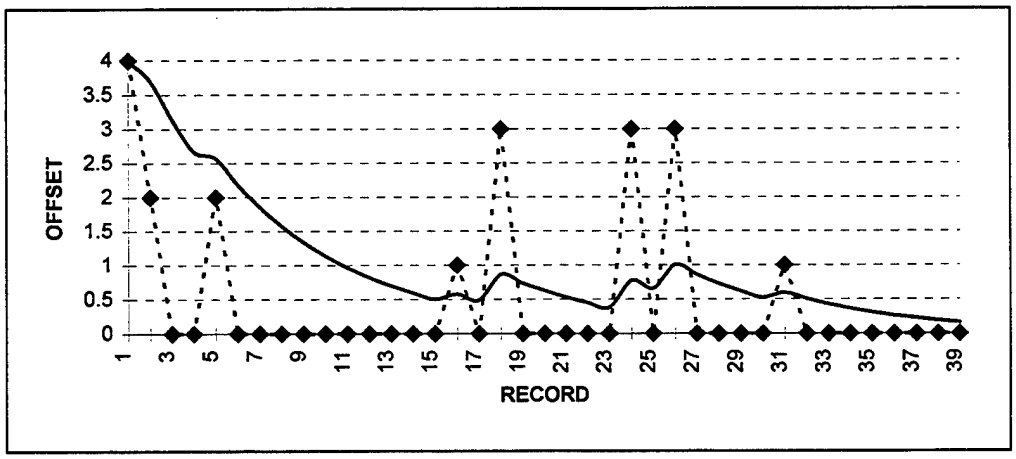

Figure 69. Time Series Results of Sample 2 for Class 5 with Weighting Scheme 20/20/20/20/20

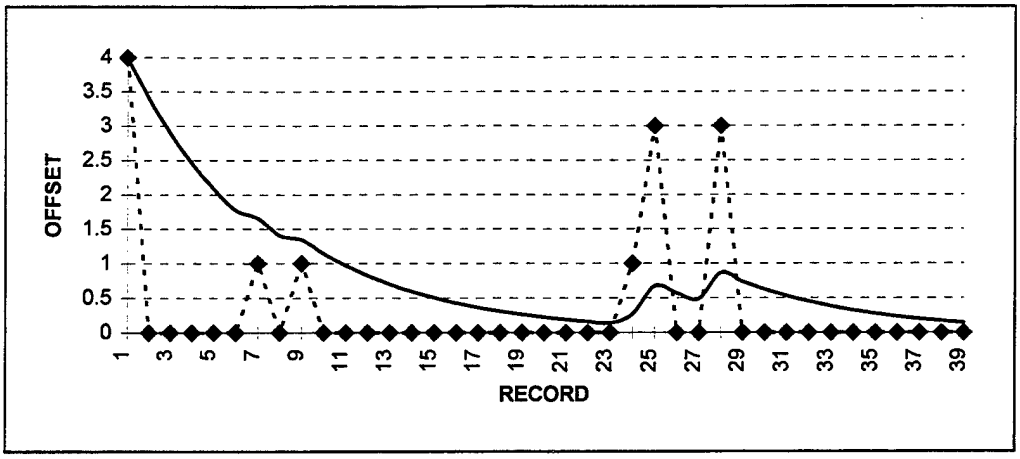

Figure 70. Time Series Results of Sample 1 for Class 5 with Weighting Scheme 30/20/30/10/10

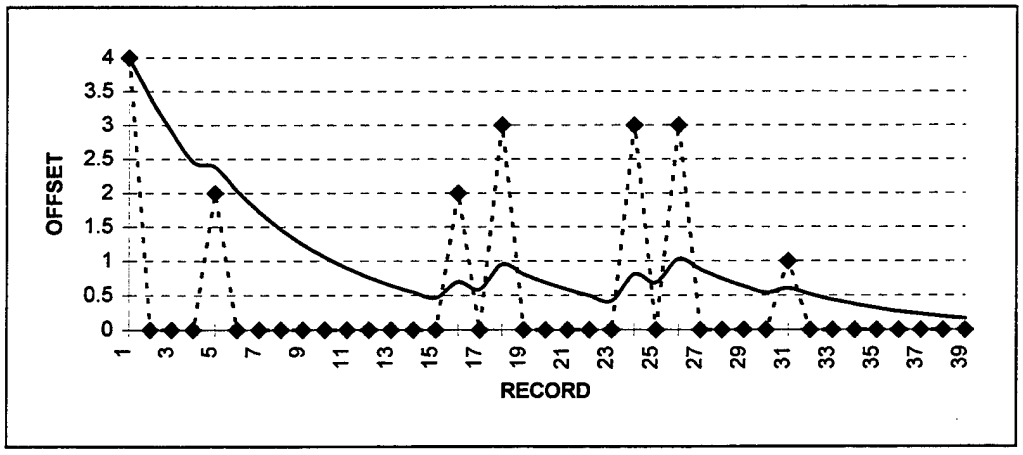

Figure 71. Time Series Results of Sample 2 for Class 5 with Weighting Scheme 30/20/30/10/10

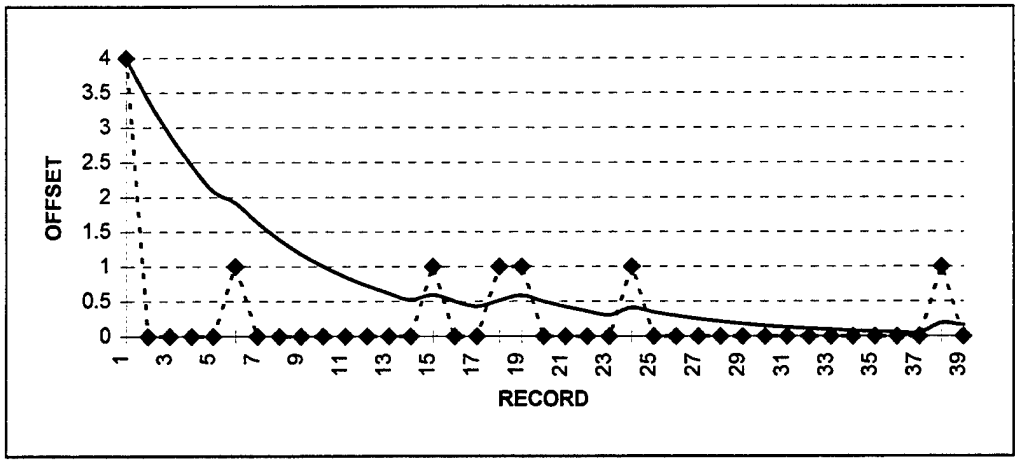

Figure 72. Time Series Results of Sample 1 for Class 5 with Weighting Scheme 50/30/00/10/10

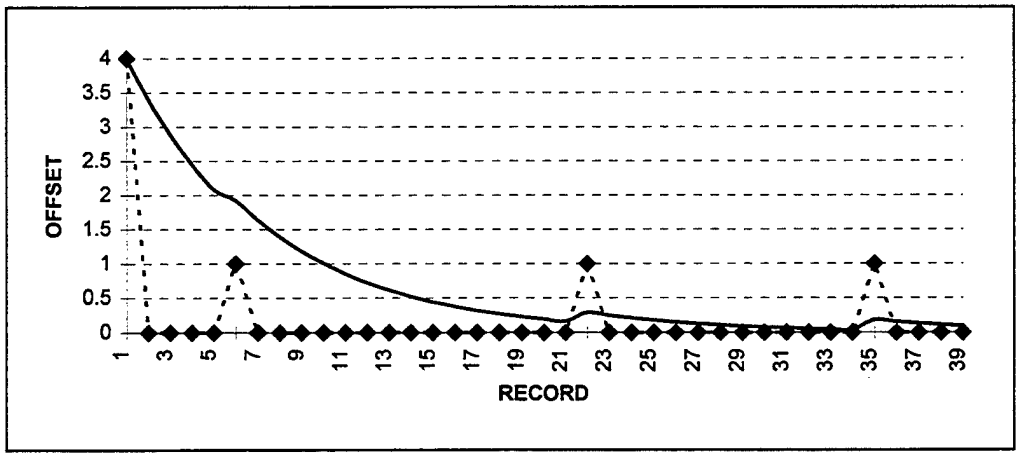

Figure 73. Time Series Results of Sample 2 for Class 5 with Weighting Scheme 50/30/00/10/10

Bibliography

Atkinson, Lee, and Mark Atkinson. Using C. Carmel IN: Que Corporation, 1990.

Bhatia, Sanjiv K., Jitender S. Deogun, and Vijay V. Raghavan. "Formation of Categories in Document Classification Systems," Lecture Notes In Computer Science, 507: 91-97 (1991).

Bolden, Bobbie and Willie Pollard. Management Assistants to the Base Records Manager, 88 CG/IMADA, WPAFB OH. Personal interview. 20 Sep 96.

Cheng, Patrick T. K. and Albert K. W. Wu. "ACS: An Automatic Classification System," Journal of Information Science, 21: 289-299 (1995).

Cosgrove, S.J. and J.M. Weimann. "Expert System Technology Applied to Item Classification," Library Hi Tech, 10: 33-40 (1992).

Department of Defense. Department of Defense Design Criteria Standard for Records Management Application: Functional Baseline Requirements. Draft DoD-STD-5015.2. Arlington VA: OASD(C3I) C3/IT, 20 May 1996.

Department of Defense Records Management Business Process Reengineering (DoD RM-BPR). Compendium Report. Arlington VA: ANDRULIS Research Corporation, August 1994.

Department of Defense Records Management Task Force (DoD RMTF). Managing Information As Records 2003. Arlington VA: ANDRULIS Research Corporation, January 1995.

Department of the Air Force. Disposition of Air Force Records: Records Disposition Schedule. AFR 4-20 Vol 2. Washington: HQ USAF, 1 May 1992.

Firebaugh, Morris W. Artificial Intelligence: A Knowledge-Based Approach. Boston: Boyd & Fraser Publishing Company, 1988.

Fox, Christopher. "Lexical Analysis and Stoplists," in Information Retrieval: Data Structures and Algorithms. Ed. William B. Frakes and Ricardo Baeza-Yates. Englewood Cliffs NJ: Prentice Hall, 1992.

Goel, Ashish. "The Reality and Future of Expert Systems," Information Systems Management, 11: 53-61 (Winter 1994).

Hayes-Roth, Frederick. "Knowledge Systems: An Introduction," Library Hi Tech, 10: 15-29 (1992).

Kendall, Kenneth E. and Julie E. Kendall. Systems Analysis and Design (Third Edition). Upper Saddle River NJ: Prentice Hall, 1995.

Larson, Ray R. "Experiments in Automatic Library of Congress Classification," Journal of the American Society for Information Science, 43: 130-148 (March 1992).

Losee, Robert M. and Stephanie W. Haas. "Sublanguage Terms: Dictionaries, Usage, and Automatic Classification," Journal of the American Society for Information Science, 46: 519-530 (August 1995).

McClave, James T. and P. George Benson. Statistics for Business and Economics (Sixth Edition). New York: Macmillan College Publishing Company, 1994.

McPharlin, Anne. AF/SCXR, Washington DC. Electronic Mail Message. 13 September 1995.

Mockler, Robert J. and D. G. Dologite. Knowledge-Based Systems: An Introduction to Expert Systems. New York: Macmillan Publishing Company, 1992.

Paice, C. D. Information Retrieval and the Computer. London: Macdonald and Jane's Publishers Ltd., 1977.

Porter, M. F. "An Algorithm for Suffix Stripping," Program, 14: 130-137 (July 1980).

Prescott, Daryll R., William Underwood, and Mark Kindl. Functional Baseline Requirements and Data Elements for Records Management Application Software. Contract DAKF11-93-C-0043. Atlanta GA: Army Research Laboratory, 28 August 1995.

Van Rijsbergen, C. J. Information Retrieval (Second Edition). London: Butterworth, 1979.

Savic, Dobrica. "Designing an Expert System for Classifying Office Documents," Records Management Quarterly, 28: 20-29 (July 1994).

Secretary of the Air Force (SECAF). Air Force Records Management Program. AFI 37-122. Washington: Secretary of the Air Force, 11 January 1994a.

-----. Management of Records. AFMAN 37-123. Washington: Secretary of the Air Force, 31 August 1994b.

-----. Records Disposition Schedule. AFMAN 37-139. Washington: Secretary of the Air Force, 1 March 1996.

Weckert, John. "Sidebar: Expert Systems," Library Hi Tech, 10: 30-32 (1992).

Vita

Captain David W. Snoddy was born 28 August 1970 in Wooster, Ohio. He graduated from Triway High School, Wooster, Ohio, in 1988. He graduated Magna Cum Laude from Kent State University, Kent, Ohio, where he received a Bachelor of Arts Degree in Technology with a minor in Psychology. Upon graduation in May 1992, he was commissioned through the Air Force Reserve Officer Training Corps. His initial assignment was as the Squadron Section Commander for the 650th Supply Squadron at Edwards Air Force Base, California, on 21 January 1993. On 1 October 1993, he became the Executive Officer/Squadron Section Commander for the newly formed 650th Logistics Group. Hand-picked by the 650th Air Base Wing Commander, he became the Executive Officer for the 650th Support Group which was activated on 1 July 1994.

In May 1995, Captain Snoddy entered the School of Logistics and Acquisition Management, Air Force Institute of Technology.

Capt Snoddy is married to the former Stacy M. Martin of Wooster, Ohio.

Permanent Address: 4409 Buss Road
Wooster, OH 44691

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1996	Master's Thesis

4. TITLE AND SUBTITLE
RECORDS ANALYSIS AND CLASSIFICATION SYSTEM: A PROOF OF CONCEPT SYSTEM FOR THE AUTOMATED CLASSIFICATION OF UNITED STATES AIR FORCE RECORDS

5. FUNDING NUMBERS

6. AUTHOR(S)
David W. Snoddy, Captain, USAF

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
2750 P Street
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GIR/LAR/96D-11

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Captain Anne McPharlin, C4I Resource Analyst
HQ USAF/SCXR
1250 Air Force Pentagon
Washington DC 20330-1250

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 Words*)

The records management process utilized within the Department of Defense (DoD) is currently labor intensive. Work is being done to automate portions of this process, but classifying documents and assigning disposition instructions remains a labor intensive, manual operation. Although the requirement for this capability was identified by a DoD sponsored study, an automated computer-based system which can classify and apply disposition instructions has yet to be developed for use within the DoD. This thesis study presents a proof of concept computer program called the Records Analysis and Classification System (RACS) which was developed to demonstrate computer-based techniques for the automated classification of official records. To demonstrate the operation of RACS, a sample of 113 records was collected from the files of an organization at Wright-Patterson AFB. An analysis of the results of the tests conducted with the RACS system indicated that it was capable of accurately classifying 72 out of the 113 records on average. Additionally, the RACS program was designed as a learning system and the test results indicated that it was in fact capable of improving its classification accuracy over time.

14. SUBJET TERMS	15. NUMBER OF PAGES
Records Management, Classification, Artificial Intelligence, Natural Language	200
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

AFIT Control Number AFIT/GIR/LAR/96D-11

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. **Please return completed questionnaire** to: AIR FORCE INSTITUTE OF TECHNOLOGY/LAC, 2950 P STREET, WRIGHT-PATTERSON AFB OH 45433-7765. Your response is **important.** Thank you.

1. Did this research contribute to a current research project? a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?
a. Yes b. No

3. **Please estimate** what this research would have cost in terms of manpower and dollars if it had been accomplished under contract or if it had been done in-house.

Man Years_____ $_____

4. Whether or not you were able to establish an equivalent value for this research (in Question 3), what is your estimate of its significance?

a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments (Please feel free to use a separate sheet for more detailed answers and include it with this form):

_____ _____
Name and Grade Organization

_____ _____
Position or Title Address