

AIM:

The main aim of this project to detect spam and unspam mails.

ABSTRACT:

Spam has been a serious and annoying problem for decades. Even though plenty of solutions have been put forward, there still remains a lot to be promoted in filtering spam emails more efficiently. Nowadays a major problem in spam filtering as well as text classification in natural language processing is the huge size of vector space due to the numerous feature terms, which is usually the cause of extensive calculation and slow classification. Extracting semantic meanings from the content of texts and using these as feature terms to build up the vector space, instead of using words as feature terms in tradition ways, could reduce the dimension of vectors effectively and promote the classification at the same time. In this project, a LDA (latent Dirichlet allocation) and SVM(Support Vector Machine)was proposed and the related feature terms were selected from the semantic meanings of the text content. Both the extraction of semantic meanings and the selection of feature terms are implemented through attaching annotations on the texts layer by- layer.

INTRODUCTION:

Spam, which is unsolicited bulk email, has packed into everyone's daily life for decades. The ballooned spam has deeply influenced the efficiency using of email as email is not only used to support conversation but also as a task manager and document delivery system and archive nowadays . Some researches even brought out a dreadful fact that all kinds of spam emails can account for as high as 88%~92% of all the emails delivered daily . The topics of spam email vary from illegal products and services to intimidation and fraud, besides spam emails usually bring about potential risks like information theft by helping plenty of malware spread with extreme rapidity. Hence, in order to prevent the situation from deteriorating many solutions have been proposed and spam filtering technologies have been developed deeply and commercialized for years. But there still are hundreds of spam emails being encountered by every email user per year, indicating that the improvement is still necessary and urgent in spam filtering. Because spam emails could be exposed in every part of the email delivering process, there are many methods. frequently used in spam filtering and always worked in conjunction, such as whitelists/blacklists, challenge-response, rule-based filtering, keyword-based filtering, content-based filtering, etc. Spam filtering can be regarded as a special binary classification task of text to determine whether an email is spam or not. The most familiar way to represent a set of texts is transform each text into a vector based on the words of it, then a vector space model of all texts in the set is formed. With such kind of vector space model, the text classification could be achieved by clustering method or machine learning algorithms easily. Machine learning algorithms are widely applied in text classification and usually perform well, among which Support Vector Machines and naive Bayes classifier are the most popular spam filters presented in the literature. However, picking words as features to create text vector always lead to huge amount of computation in the process of classification, because of the high dimension of vector space model

built up, especially when large volume of texts are involved which is familiar in spam filtering nowadays. Although plenty of feature selection methods have been proposed in the literature , the extensive feature terms still are the major problem in text classification and spam filtering so far. This is another weakness of spam filters requiring improvement.in our proposed system, we implement two algorithms, LDA and SVM algorithm. user register their own details and login. send the mail to particular mail id with using keywords like invoice, discount it will consider as spam mail and stored as spam. without using keyword will consider as normal mail. upload the datasets for any algorithm .view the datasets .if other's can't see our data's. we encrypt the data's. Then we show inbox, spam and sent messages

LITERATURE SURVEY:

1. Machine Learning for E-mail Spam Filtering: Review, Techniques and Trends

Alexy Bhowmick:

We present a comprehensive review of the most effective content-based e-mail spam filtering techniques. We focus primarily on Machine Learning-based spam filters and their variants, and report on a broad review ranging from surveying the relevant ideas, efforts, effectiveness, and the current progress. The initial exposition of the background examines the basics of e-mail spam filtering, the evolving nature of spam, spammers playing cat-and-mouse with e-mail service providers (ESPs), and the Machine Learning front in fighting spam. We conclude by measuring the impact of Machine Learning-based filters and explore the promising offshoots of latest developments.

2. Incremental personalized E-mail spam filter using novel TFDJR feature selection with dynamic feature update

Gopi Sanghani:

Communication through e-mails remains to be highly formalized, conventional and indispensable method for the exchange of information over the Internet. An ever-increasing ratio and adversary nature of spam e-mails have posed a great many challenges such as uneven class distribution, unequal error cost, frequent change of content and personalized context-sensitive discrimination. In this research, we propose a novel and distinctive approach to develop an incremental personalized e-mail spam filter. The proposed work is described using three significant contributions. First, we applied a novel term frequency difference and category ratio based feature selection function TFDJR to select the most discriminating features irrespective of the number of samples in each class. Second, an incremental learning model is used which enables the classifier to update the discriminant function dynamically. Third, a heuristic function called selectionRankWeight is introduced to upgrade the existing feature set that

determines new features carrying strong discriminating ability from an incoming set of e-mails. Three public e-mail datasets possessing different characteristics are used to evaluate the filter performance. Experiments are conducted to compare the feature selection efficiency of TFDCR and to observe the filter performance under both the batch and the incremental learning mode. The results demonstrate the superiority of TFDCR as the most effective feature selection function. The incremental learning model incorporating dynamic feature update function overcomes the problem of drifting concepts. The proposed filter validates its efficiency and feasibility by substantially improving the classification accuracy and reducing the false positive error of misclassifying legitimate e-mail as spam.

3. SPAM AND EMAIL DETECTION IN BIG DATA PLATFORM USING NAIVES BAYESIAN CLASSIFIER

G.Vijayasekaran:

Email spam is operations which are sending the undesirable messages to different email client. E-mail spam is the very recent problem for every individual. The e-mail spam is nothing it's an advertisement of any company/product or any kind of virus which is receiving by the email client mailbox without any notification. To solve this problem the different spam filtering technique is used. The spam filtering techniques are used to protect our mailbox for spam mails. In this project, we are using the Naives Bayesian Classifier with three layer framework that includes obfuscator, classifier and anomaly detector for spam classification for bulk emails. The Naïve Bayesian Classifier is very simple and efficient method for spam classification. Here we are using the real time dataset for classification of spam and non-spam mails. The feature extraction technique is used to extract the feature in terms of digest based on bucket classification. The result is to increase the accuracy of the system. And implement Self Acknowledgeable Intranet Mail System has been designed and implemented to benefit the sender about the status of his mail.

Once a mail is sent, the sender can know the receiver activity in the mail system until the mail is viewed. Finally provide the pop up window to identify the mail content at the time of open the spam mails

SYSTEM ANALYSIS:

EXISTING SYSTEM:

- In the traditional ways of natural language processing, text is always cut into small terms based on words. For every term the value of the **Term Frequency (TF)** is calculated and all the terms with their TF values together could form a vector which can represent the text.
- This method is usually described as representing the text by turning it into a bag of words. Obviously, when there are abundant number of words or terms involved the size of the vector space formed can be incredibly huge.
- Nowadays, this representation method is still familiar in natural language processing, but many term selection techniques have developed to reduce the size of vector space. For example, the value of **Inverse Document Frequency (IDF)** is proposed to be combined with TF in order to take away the terms which barely influence the classification.
- The **TF-IDF** method as a purely statistical method may not reflect the importance of every term for classification correctly and hardly reflect the information about structures of all the terms in text at all, which lead to an obvious **disadvantage**. This is because in the art of **natural language** the flexible and ingenious combination of words is **unpredictable**.
- **Latent Semantic Analyzing(LSA)** was first put, which was based on the TF-IDF method and designed for distinguish the synonyms in information retrieval at first, then applied to semantic recognition and text classification later as LSA could evaluate the similarity between concepts, sentences and texts .

DISADVANTAGES:

- Cannot use dataset.

- Cannot encrypt the datas.

PROPOSED SYSTEM:

- Spam Filtering is to find out whether the emails are spam or not. In our experiment, the class of personal letter was the only class that was defined as ham.
- But for a specific user, the recruitment messages could also be important and useful, as a consequence, the emails in the class of recruitment should be identified as ham for this user.
- So with the multiple classification the personalized spam filtering system was much more easier to develop.
- In the Proposed System we have implemented the **LDA and SVM** Algorithm. If the Email is having following attributes means, that email is a spam Attributes are, **Invoicing, training, recruiting, eroticism, website, selling, letter, defrauding, Etc.**
- We have created the web application as like gmail. By using the LDA and SVM Algorithm, spam mails are filtered based on the category. Accuracy for both the algorithms is calculated. And finally it proved that **LDA has higher accuracy than the SVM Algorithm.**

ADVANTAGES:

- Use dataset
- Encrypt the datas

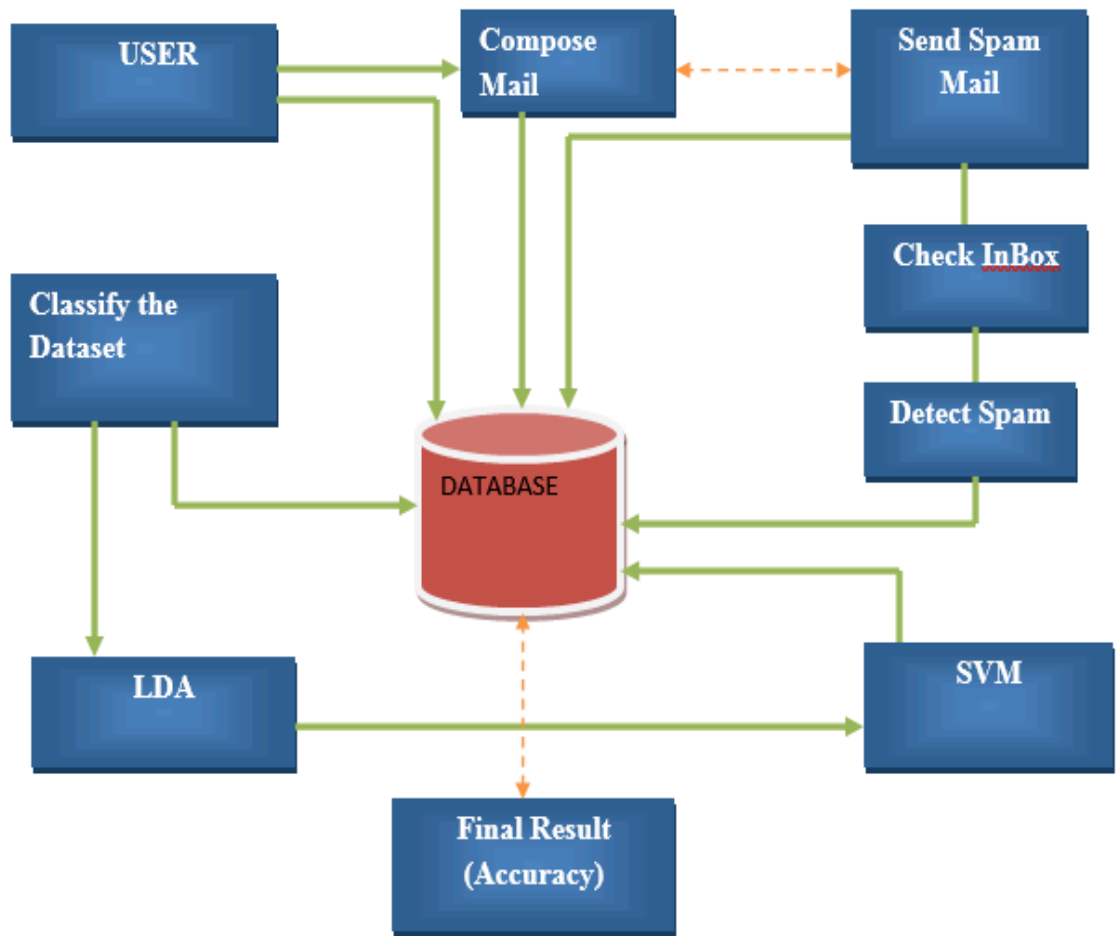
LDA(Latent Dirichlet allocation):

In natural language processing, latent Dirichlet allocation (**LDA**) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar.

SVM(Support Vector Machine):

“Support Vector Machine” (**SVM**) is a supervised machine learning **algorithm** which can be used for both classification or regression challenges. However, it is mostly used in classification problems. ... Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

SYSTEM ARCHITECTURE:



IMPLEMENTATION:

In this project we have :

1)User

- **Register**
- **Login**
- **Compose**
- **Inbox**
- **Sent**
- **Spam**
- **Accuracy**

1)User

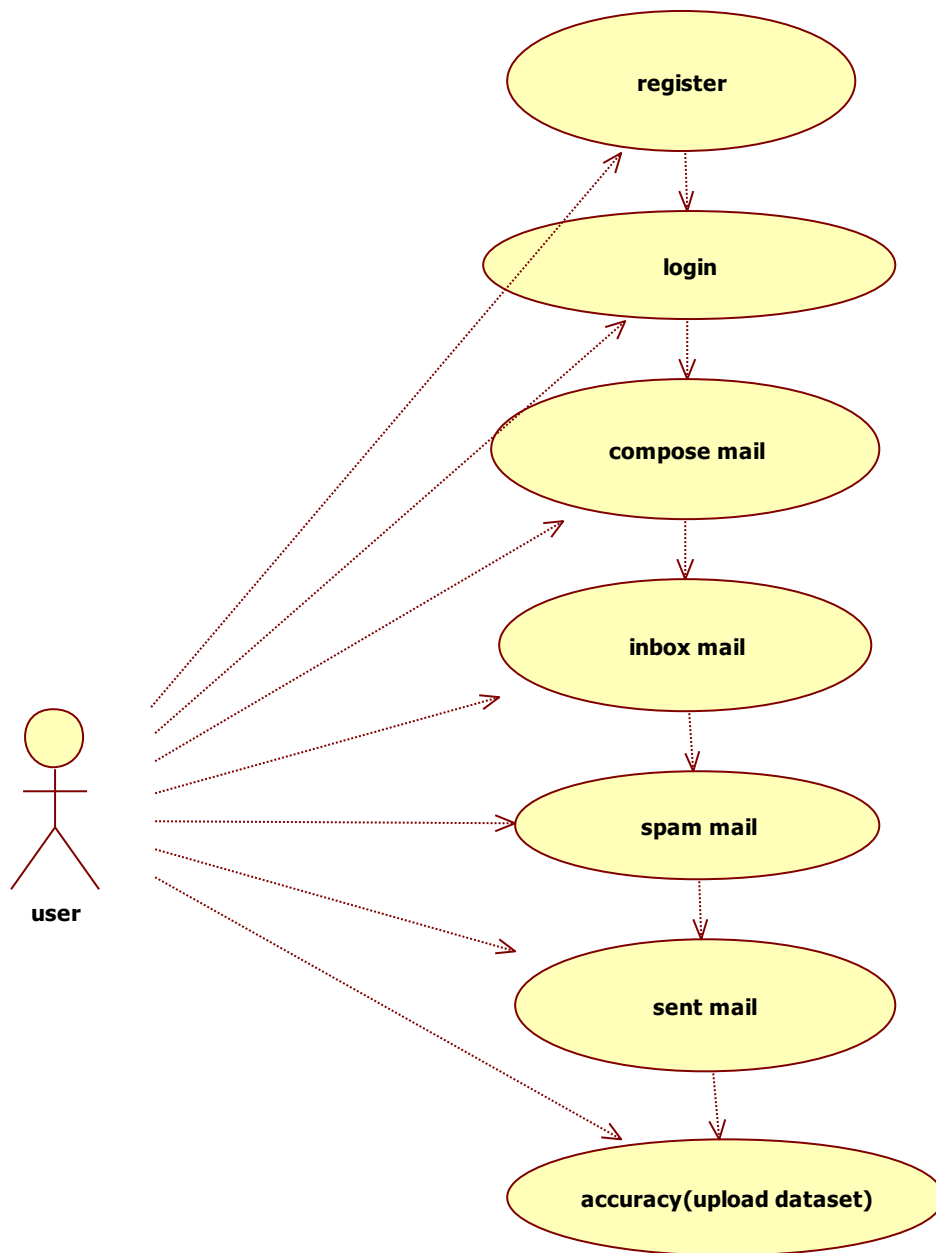
- **Register**
User register their own details
- **Login**
User login
- **Compose**
User send the mail without using keywords, with using keywords
1)without using keywords, send as normal mail
2)with using keywords, send as spam mail
- **Inbox**
User view normal mail details
- **Sent**
User view sent mail details
- **Spam**

User view spam mail details

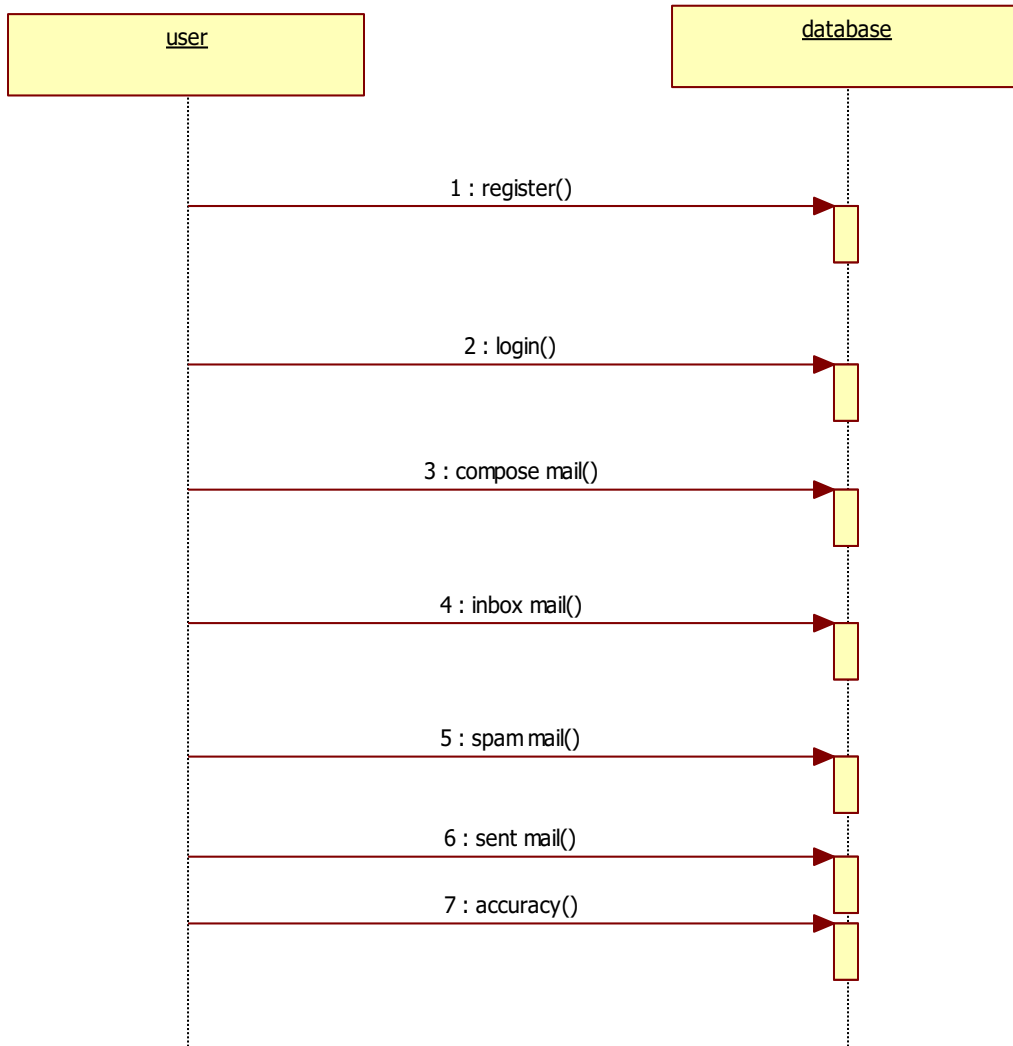
- **Accuracy**

User upload dataset for any one algorithm, show the dataset details. If other's can't see dataset details, we encrypt the data's.

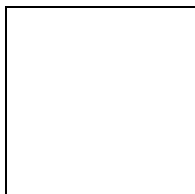
USECASE DIAGRAM:



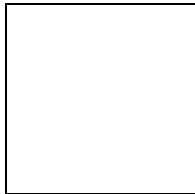
SEQUENCE DIAGRAM:



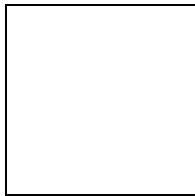
COLLABORATION DIAGRAM:



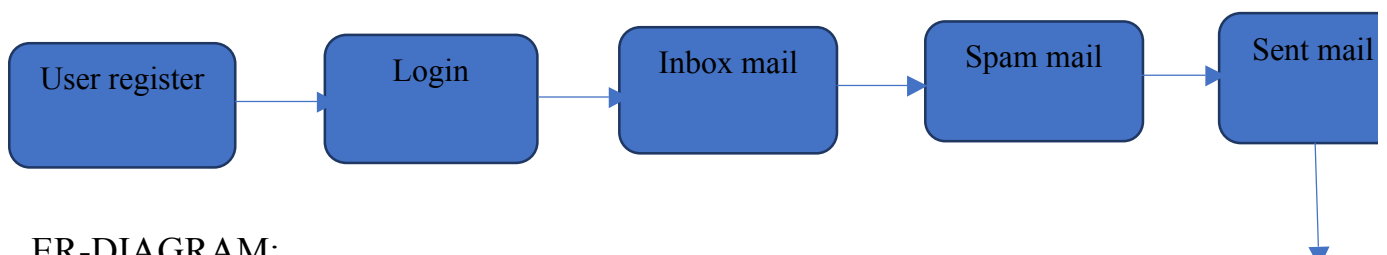
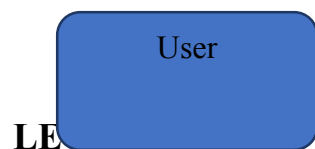
DEPLOYMENT DIAGRAM:



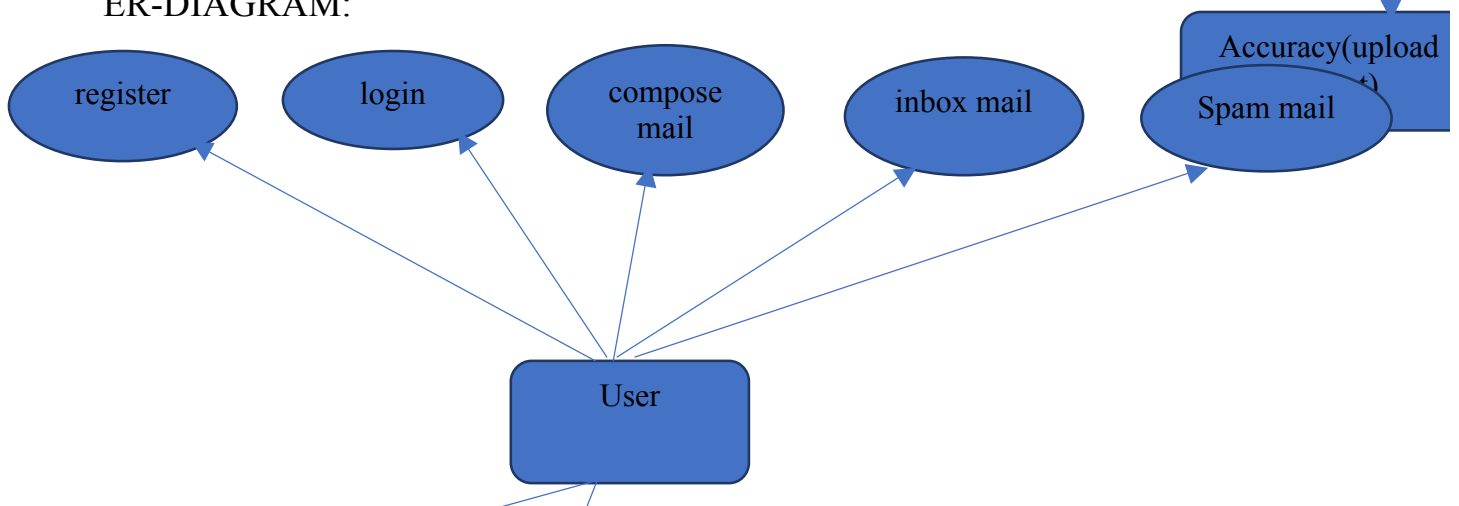
CLASS DIAGRAM:



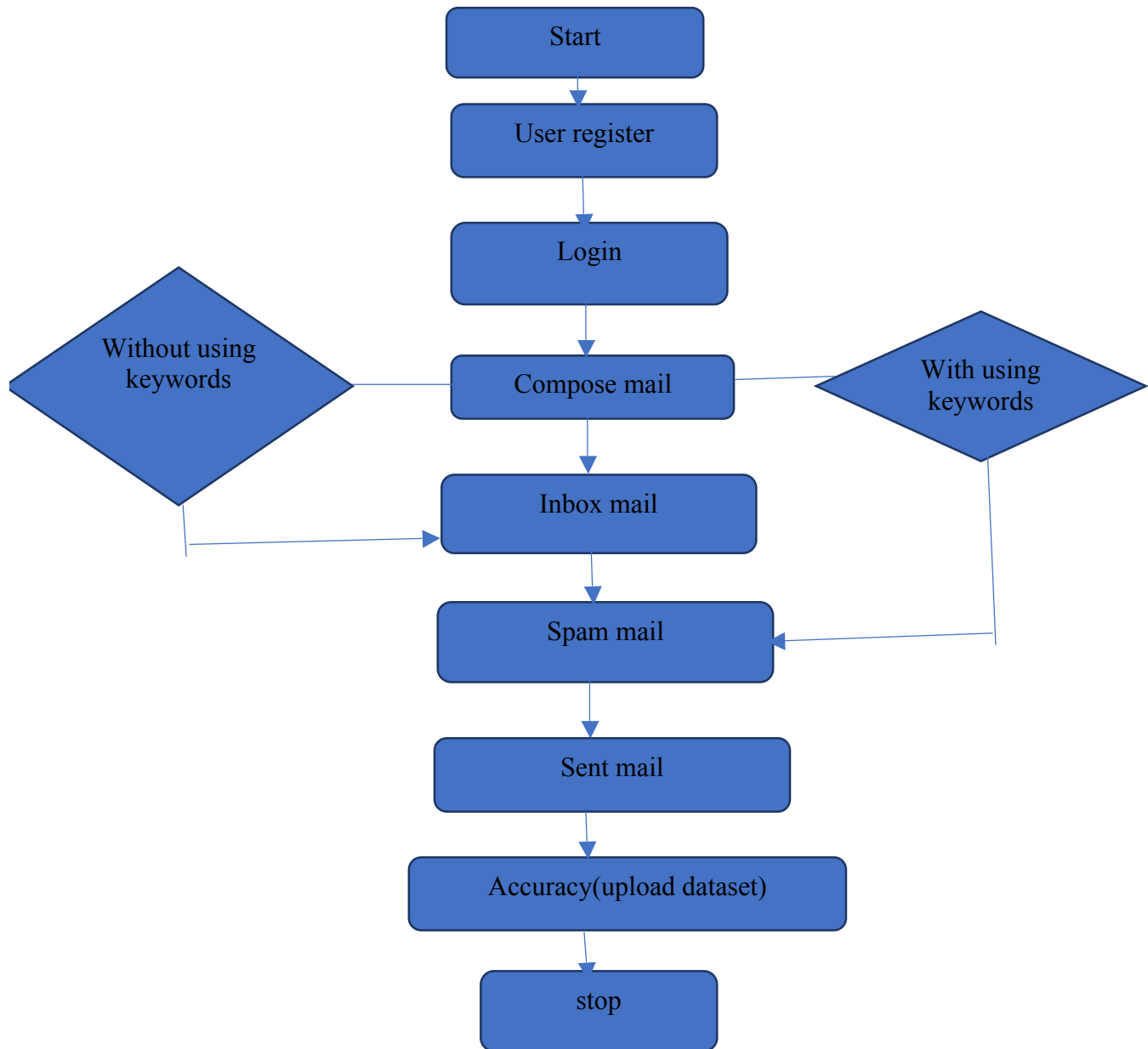
DFD:DATA FLOW DIAGRAM
LEVEL-0 DFD:



ER-DIAGRAM:



ACTIVITY DIAGRAM:



Software Environment

Java Technology

Java technology is both a programming language and a platform.

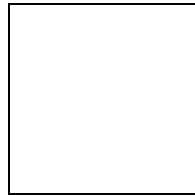
The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes*—the platform-independent codes interpreted by the interpreter on the Java

platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



The Java Platform

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be

described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

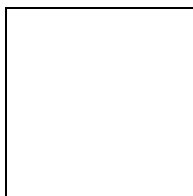
The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, *What Can Java Technology Do?* Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

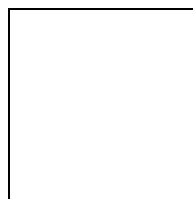
An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials:** Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets:** The set of conventions used by applets.

- **Networking:** URLs, TCP (Transmission Control Protocol), UDP (User Datagram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization:** Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security:** Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components:** Known as JavaBeans™, can plug into existing component architectures.
- **Object serialization:** Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™):** Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and

requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly:** Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
- **Write less code:** Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code:** The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop programs more quickly:** Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java:** You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure JavaTM Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write once, run anywhere:** Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.

- **Distribute software more easily:** You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each

maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. ***SQL Level API***

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

2. ***SQL Conformance***

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

3. ***JDBC must be implemental on top of common database interfaces***

The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

4. ***Provide a Java interface that is consistent with the rest of the Java system***

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

5. ***Keep it simple***

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

6. *Use strong, static typing wherever possible*

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

7. *Keep the common cases simple*

Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally we decided to proceed the implementation using Java [Networking](#).

And for dynamically updating the cache table we go for MS [Access](#) database.

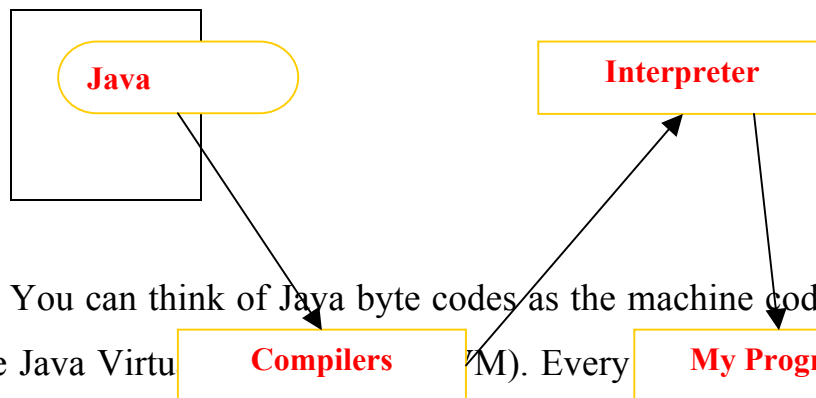
Java ha two things: a programming language and a platform.

Java is a high-level programming language that is all of the following

Simple	Architecture-neutral
Object-oriented	Portable
Distributed	High-performance
Interpreted	multithreaded
Robust	Dynamic
Secure	

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.



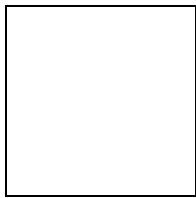
You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (JVM). Every **My Program** whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

Networking

TCP/IP stack

The TCP/IP stack is shorter than the OSI one:



TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

IP datagram's

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer

supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

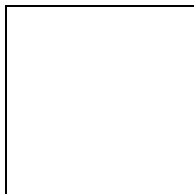
Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

Total address



The 32 bit address is usually written as 4 integers separated by dots.

Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call `socket`. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int family, int type, int protocol);
```

Here "family" will be `AF_INET` for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

JFree Chart

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFreeChart is "open source" or, more specifically, [free software](#). It is distributed under the terms of the [GNU Lesser General Public Licence](#) (LGPL), which permits use in proprietary applications.

1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

2. Time Series Chart Interactivity

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series

data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

4. Property Editors

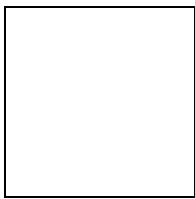
The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

J2ME (Java 2 Micro edition):-

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing

mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

1. General J2ME architecture



J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the The profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define

certain uses for devices. We'll cover profiles in depth in the The following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

2.Developing J2ME applications

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role preverification plays in this process.

3.Design considerations for small devices

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

- * Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.

- * Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

- * Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

4.Configurations overview

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

- * **Connected Limited Device Configuration (CLDC)** is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

* **Connected Device Configuration (CDC)** is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

5.J2ME profiles

What is a J2ME profile?

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

Profile 1: KJava

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT).

However, because it is not a standard J2ME package, its main package is `com.sun.kjava`. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

Profile 2: MIDP

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application

development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

- * `java.lang`
- * `java.io`
- * `java.util`
- * `javax.microedition.io`
- * `javax.microedition.lcdui`
- * `javax.microedition.midlet`
- * `javax.microedition.rms`

OUTPUT

BY USING SVM-

Mail Detection

×

!come Vb

Compose

Inbox

Spam

Sent

Accuracy

Logout

Result for Collaborative Filtering

{Genuine=138, recruiting=1, defrauding=3, selling=6, training Course=1}

Precision Value is : **1.0**

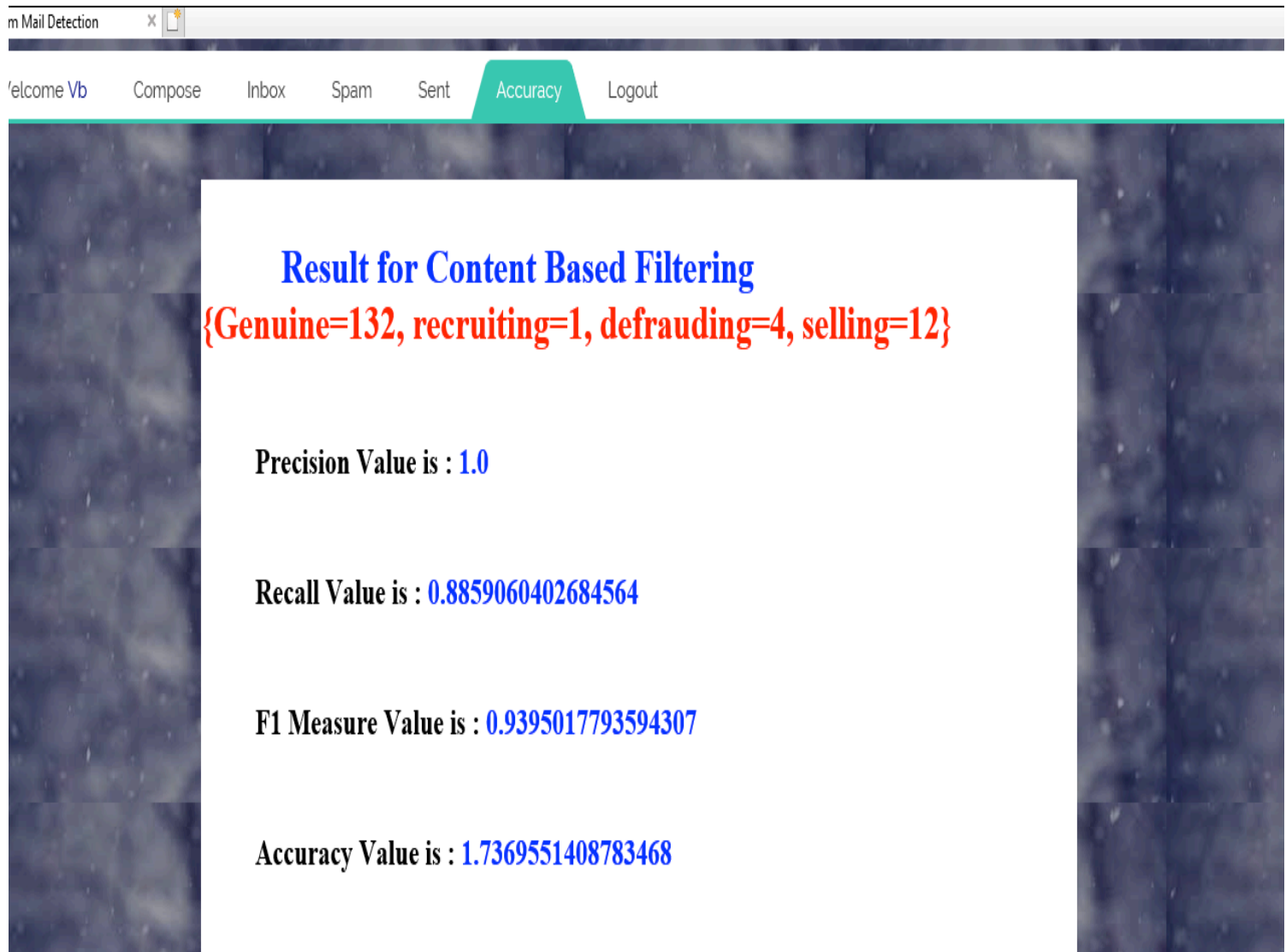
Recall Value is : **0.040268456375838924**

F1 Measure Value is : **0.07741935483870968**

Accuracy Value is : **0.8718345116502775**

OUTPUT

BY USING LDA-



The screenshot shows a web application interface with a browser window titled "m Mail Detection". The application has a navigation bar with links: "Welcome Vb", "Compose", "Inbox", "Spam", "Sent", "Accuracy" (highlighted in green), and "Logout". The main content area displays the following results:

Result for Content Based Filtering
{Genuine=132, recruiting=1, defrauding=4, selling=12}

Precision Value is : 1.0

Recall Value is : 0.8859060402684564

F1 Measure Value is : 0.9395017793594307

Accuracy Value is : 1.7369551408783468

CONCLUSION:

Because of the characters of spam issues, massive and continuous, spam filtering approaches with higher performance are still required to be developed urgently. We proposed a LDA latent Dirichlet allocation and SVM(support vector machine) with text classification method based on the semantic information.

FUTURE WORK:

There are plenty further work that can proceed based on the semantics-based text classification because of the highly effective and flexible method of feature selection. For example, the application of online SVM is limited because of the low training speed, but associated with the semantics based feature terms selection by attaching annotations would highly promote the efficiency of online SVM, which is also promising in online spam filtering. Moreover, the method of attaching annotations based on semantics has great potential in text classification in the situation where multiple languages are involved, since this method focus mainly on the semantic meanings other than the words. It also can play an efficient role in classification of SMS messages, news, scientific literature and the messages delivered through social networks. Besides, the convenience in realizing the personalized spam filtering is significance for user-friendly and approachable commercial spam filtering system.

REFERENCES:

- [1] T. A. Almeida, A. Yamakami, and J. Almeida, "Filtering Spams using the Minimum Description Length Principle," Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 1854–1858, 2010.
- [2] C. Kreibich, et al. "Spamcraft: An Inside Look At Spam Campaign Orchestration," Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more, pp. 4–4, 2009.
- [3] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," Proceedings of the 2006 Conference on Human Factors in Computing Systems, 2006, pp. 581–590.
- [4] A. Bergholz, J. D. Beer, et al., "New filtering approaches for phishing email," Journal of Computer Security, 2010, 18(1), pp. 7–35.
- [5] C. Kanich, C. Kreibich, et al., "Spamalytics: an empirical analysis of spam marketing conversion," Proc of Ccs, 2008, 52(9), pp. 99–107.
- [6] A. Dasgupta, M. Gurevich, and K. Punera, "Enhanced Email Spam Filtering through Combining Similarity Graphs," WSDM, 2011, pp. 785–794.
- [7] H. Drucker, S. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," Neural Networks IEEE Transactions on, 1999, 10(5), pp. 1048–1054.
- [8] A. K. J. Alspector, "SVM-based filtering of e-mail spam with content-specific misclassification costs," Proc of Textdm01 Workshop on Text Mining, 2001.
- [9] J. M. G. Hidalgo, "Evaluating cost-sensitive unsolicited bulk email categorization," ACM Symposium on Applied Computing, 2002, pp. 615–620.
- [10] I. Androutsopoulos, G. Paliouras, and E. Michelakis, "Learning to filter unsolicited commercial e-mail," International Proceedings of Computer Science & Information Tech, 2004, 10(1), pp. 4324–4330.