

Molei Hong

AI PROGRAMMING WITH JAVA

Final Thesis

CENTRIA UNIVERSITY OF APPLIED SCIENCES

Information Technology

December 2020

Centria University of Applied Sciences	Date 1/9/2021	Author Molei Hong
Degree programme Information Technology		
Name of thesis AI PROGRAMMING WITH JAVA		
Instructor Kauko Kolehmainen	Pages 42	
Supervisor Kauko Kolehmainen		
<p>The thesis was about artificial intelligence project in Java. It started with the theoretical part of basic presentation of Java programming language, introduced its history, importance and features. The significant description of Java platform component Java Virtual Machine was involved. The definition of AI, the types of AI and the different AI level which included machine learning and deep learning were also presented. In the practical part, Weka, a software of machine learning algorithm in Java was introduced and there were two data mining and classification experiments by using Weka demonstrated. After that, a Java project with Weka's Naïve Bayesian algorithm illustrated the principle of email classification. In the project, necessary elements such as Weka packages, the datasets and important methods were presented in detail.</p>		

<p>Key words Java programming, Java Virtual Machine, Artificial Intelligence, Weka, Naïve Bayesian, spam filter, data classification.</p>
--

ABSTRACT
CONCEPT DEFINITIONS
CONTENTS

1 INTRODUCTION.....	1
2 JAVA PROGRAMMING LANGUAGE.....	2
2.1 Java platform Standard Edition	Error! Bookmark not defined.
2.2 Java Virtual Machine	Error! Bookmark not defined.
3 ARTIFICIAL INTELLIGENCE.....	7
3.1 The definitions of AI.....	9
3.2 The categories of AI.....	11
3.2.1 Machine Learning.....	12
3.2.2 Deep Learning.....	14
3.3 Machine Learning tools and libraries in Java	15
4 WEKA	16
4.1 The explorer experiment	18
4.2 The class structure Weka	23
4.2.1 The Weka.core package	24
4.2.2 The Weka.classifiers package	25
5 PROJECT EXPLANATION	26
5.1 Naive Bayesian algorithmic reasoning	26
5.2 Selection of datasets.....	28
5.3 Importing Weka Packages	30
5.4 Training data	32
5.5 Testing data	35
5.6 Evaluation result.....	36
6 CONCLUSION	38
REFERENCES	39
SOURCE OF FIGURES	41

LIST OF FIGURES

Figure 1. Benefits of programming AI in Java	3
Figure 2. The interface of Java Platform Manager (Netbeans IDE 8.2)	4
Figure 3. The internal architecture of the JVM.....	5
Figure 4. The structure of Java 2 platform SE 5.0.....	6
Figure 5. Four categories are organized for the definitions of AI.	10
Figure 6. Autonomous driving as the start of Artificial Intelligence level	11
Figure 7. Three learning methods of machine learning algorithms	13
Figure 8. Weka GUI Chooser (Version 3.8.3)	17
Figure 9. Problems of ARFF format in UCI dataset.....	18
Figure 10. Reading in the diabetes data	18
Figure 11. Classifier output of diabetes dataset	19
Figure 12. Images of Rhino and Brontosaurus	20
Figure 13. Equal frequency, 43 examples of each in the dataset	20
Figure 14. Additional features were added to the dataset	21
Figure 15. Classifier output	22
Figure 16. Package Manager in Weka	23
Figure 17. Method of Copyable interface.....	24
Figure 18. Method distributionForInstance()	25
Figure 19. Training dataset and testing dataset.....	28
Figure 20. TrainingData.arff opened as txt file.....	29
Figure 21. Libraries included weka.jar and JDK 1.8	30
Figure 22. Weka.core packages used in the source code	31
Figure 23. Weka.classifiers package and StringToWordVector class are imported in the source code.....	31
Figure 24. Defined instances, attributes of training data and locations of training dataset files ..	32
Figure 25. Attributes "label" and "text" were added and "label" was assigned values	32
Figure 26. Loading and saving dataset content.....	33
Figure 27. Converting text to feature vector	33
Figure 28. Using tokenizer to split the string	33
Figure 29. Building the classifier	34
Figure 30. Evaluating the classifier model	35
Figure 31. Main method for running classifier	36
Figure 32. Output after running.....	37

1 INTRODUCTION

In recent years, machine learning, data science and artificial intelligence have been the most talked about technologies. This is a matter of course. These advancements in technology have brought automation and business processes to a new level. Organizations of all sizes have invested millions of dollars in research and personnel to develop these extremely powerful data-driven applications. There are many different programming languages that can be used to develop machine learning and data science applications. Although Python and R have become the first choice for developing these programs, many organizations are turning to Java development to meet their needs. From enterprise-level business solutions and navigation systems to mobile phones and applications, Java is suitable for almost every technical field.

However, there are also several flaws of using Java as programming language for AI. Compared with C++, Java has lower execution speed and needs more response time. Although Java is portable and supports running on multiple platforms, when it is executed on older platforms, significant adjustments of both software and hardware are demanded. As an immature AI language, Java is under development which indicates that it is not the mainstream programming language of AI. Artificial intelligence is related to the development of artificial neural networks, search algorithms and genetic programming.

The project which author makes is a spam-filter program mainly based on machine learning algorithm Naive Bayes, which is a typical statistical-based spam filtering technology. The basis of its theory is to analyze the common keywords of a large number of spam emails to obtain a statistical model of their distribution, and to estimate the probability that the target is spam. According to the set threshold to determine whether to accept mail, its biggest feature is the self-learning function, by continuous self-updating filtering rules and ensuring long-term stable filtering efficiency without maintenance.

The main goal of this research was to implement the text data classification through training and testing the machine learning model by Netbeans integrated development environment for certifying the feasibility and effectiveness of artificial intelligence programming using the Java programming language. The limitation of this study is that it cannot be compared with the performance of projects programmed by other programming languages. Therefore, it is difficult to show the shortcomings of Java in artificial intelligence programming.

2 JAVA PROGRAMMING LANGUAGE

Java was developed by the Sun Microsystems Corporation in the 90s, widely used in personal computers, data centers, game consoles, supercomputers, mobile phones and the Internet. Maintainability, portability and transparency are main advantages of Java programming language. Java was an effort to implement the various function of object-oriented programming. Compared with the early object-oriented languages such as Common Lisp Object System, Flavors and Smalltalk developed by the AI community, Java obtains significant attention from software engineering developers. (PART IV: AI programming in Java Page 269.)

As a matter of fact, Java is widely used in real world: in Android phone, any application is written by Java. In financial services industry, Java is taken as the developing language for the front-end and back-end trading systems of global investment banks such as Citygroup, Standard Chartered and Goldman Sachs. Java also plays a big role in the web application space, the government departments like education, defense, healthcare, using Java to establish their official web applications. Powerful software and integrated development environment developed in Java e.g. IntelliJ IDEA, Eclipse and NetBeans IDE. In the embedded system, Java platform shows its important feature of “portable”, it is initially designed for embedded devices and is implementing step by step. However, Java takes the second position of the popularity ranking of programming languages. The popularity of Java programming language shows a decrease trend in the year of 2020. Because of the widespread using of Python and the dominant position of C language in software development, the usage rate of Java has declined 4.32% since 2019. (TIONE Index 2020.)

The hinge characteristic of Java is that it is an open standard with publicly available source code. Sun Microsystems manages the Java language and concerning products, however, Sun's flexible licensing policy stimulates the use of Java as a standard in the Internet community. Developers can download all the tools they needed to develop and run Java applets and applications for free from Sun's Java website. (Austerlitz 2003.)

Selecting the appropriate programming language is an important task for AI construction. There are diverse and general options such as python, C++ and LISP. Nevertheless, Java plays an essential role in AI programming. Java is the most widely used programming language in the world, it is object-oriented and scalable, which is necessary for AI projects. Gigantic Java code bases are exploited by public and private organizations and sectors, and the Java Virtual Machine as a compute environment is heavily relied. Java Virtual Machine allows developers to build a single application version that could run on all Java-enabled computing platforms. Maintainability, portability and transparency are the three main advantages of Java programming language. The following figure shows the advantages that Java has in the AI programming. (Nicholson 2020.)

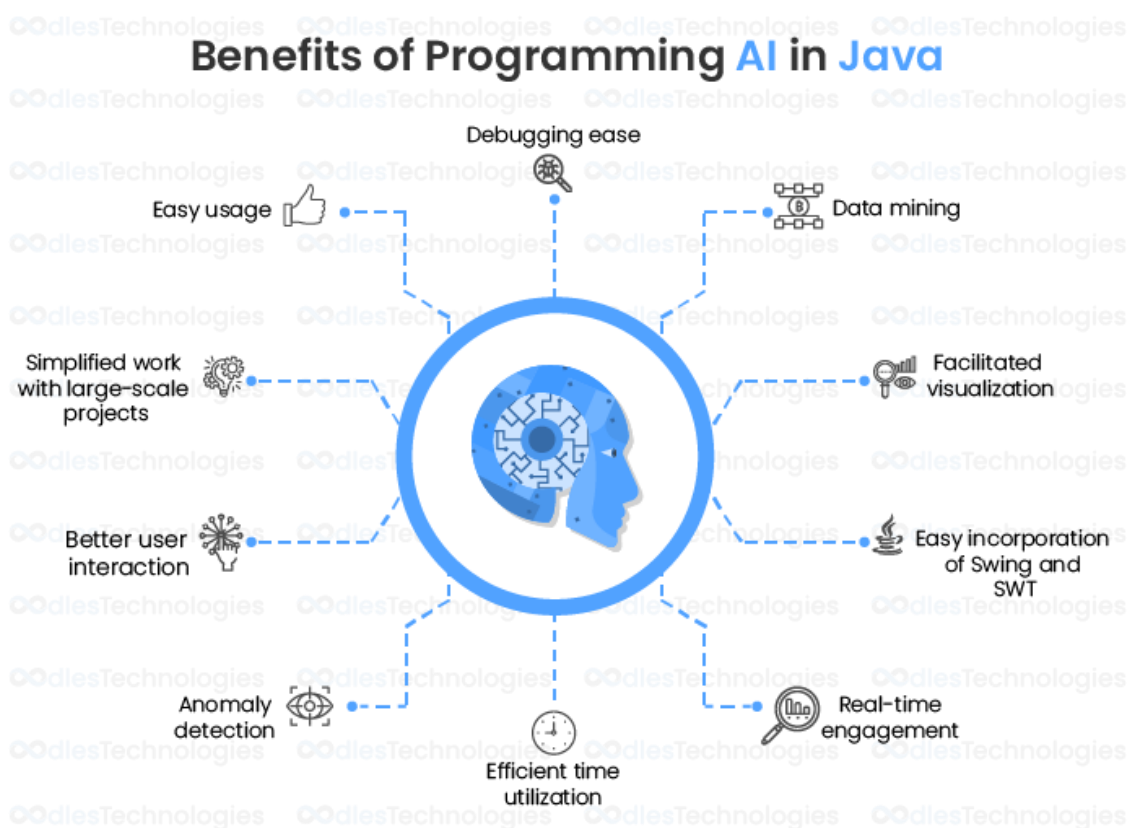


Figure 1. Benefits of programming AI in Java (Source: <https://www.oodlestechnologies.com/blogs/using-java-in-artificial-intelligence-programming/>)

2.1 Java platform standard edition

Java platform standard edition provides the Java software developing and running environment for users. Java applications with security and portability could be set up and developed rapidly on the Java platform standard edition, simultaneously, the applications could be run on the server and desktop systems of most operating systems. Java platform standard edition contains classes that form the core of the Java programming language. For example: database connection, interface definition, input/output, network programming. The following figure shows the interface of Netbeans' platform manager. (Oracle 2020.)

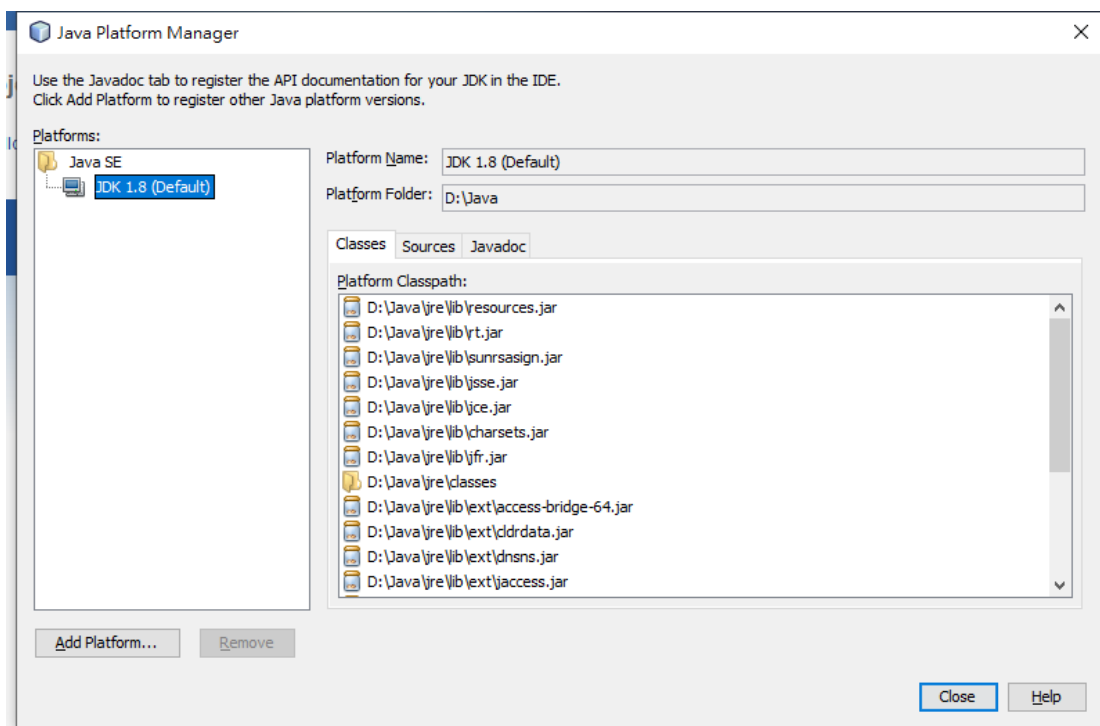


Figure 1. The interface of Java Platform Manager (Netbeans IDE 8.2, 2020)

The Java Virtual Machine (JVM) is an abstract machine which forms the foundation of Java platform. It is the specification of software that executes programs and provides the runtime environment for the program, which permits Java program to run on any operating system. Another function of JVM is to manipulate and optimize the program memory. Before Java programming language was released, all programs were stored in the specific operating systems, which means developers need to manage memory manually. There is an instruction set placed inside the JVM for controlling each memory area at runtime, that makes the allocation and relocation of the memory automatic. (Venners 1998.)

2.2 Java Virtual Machine

The JVM is a stack-based virtual machine, it has the heap store, the stack, the method area, runtime constant pools and the PC register as runtime data areas, which refer to the basic structures. A binary format which is independent from hardware and operating system is used for notating compiled code to be executed by the JVM, stored in the “class file” format. Each JVM has a class loader subsystem for loading types specified eligible names, and an execution engine for executing the instructions included in the methods of loaded classes. The internal architecture of the JVM is shown in figure 3. (Oracle 2020.)

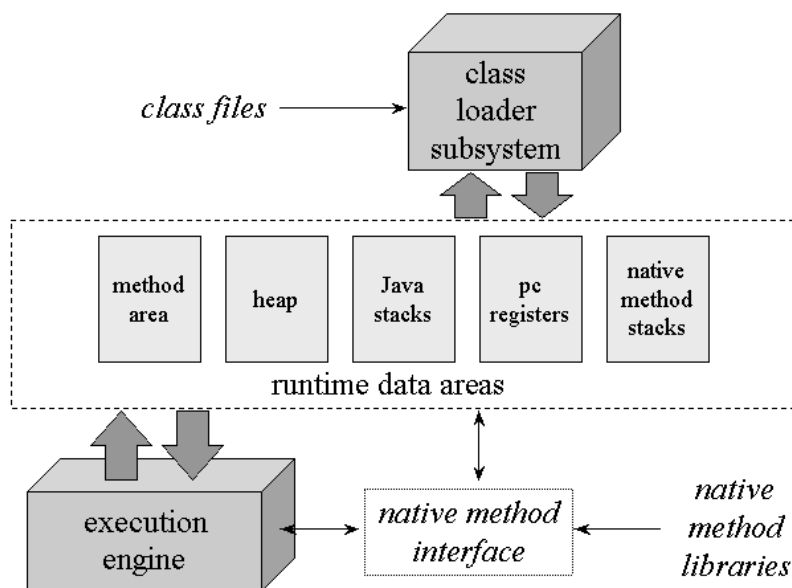


Figure 2. The internal architecture of the JVM (Source: Venners,B. Inside the Java Virtual Machine, Chapter 5, figure 5-1)

The JVM is a specification which implementors choose to provide algorithm and it is also a software program that meet the requirements of the JVM specification. Moreover, it is the runtime instance created by developers when they type the command to run the java class. The JVM is extensively deployed and heavily used in the public and private sector and it runs Java class files in a portable method. No matter what kind of hardware or operating system is used, a controllable environment for developers could be set up by the JVM. Initially the JVM was only used for Java programming language, now it has advanced to support various programming languages such as Scala, Groovy and Kotlin. And it has a predictable advantage in the further development of developing environment. (Venners 1998.)

The JDK, the JRE and the JVM shape the core technology of Java programming. The JVM is the key component of Java platform that execute programs, the JRE contains the Java class library and forms the JVM, the JDK helps programmers to develop Java application that could be executed by the JVM and run by the JRE. Figure 4 illustrates the structure of Java 2 platform, which indicates that JVM is included in the Java Runtime Environment within Java development kit. (Parahar 2020.)

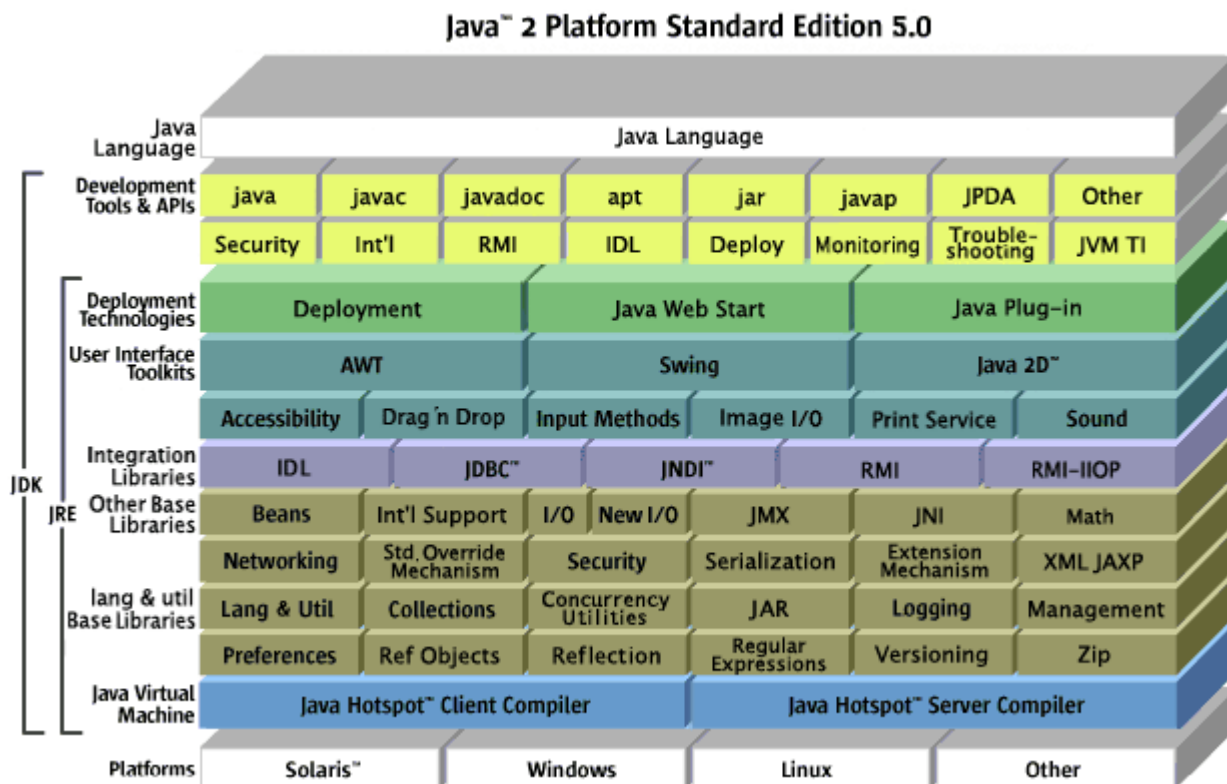


Figure 3. The structure of Java 2 platform SE 5.0 (Source: <https://docs.oracle.com/javase/1.5.0/docs/>)

3 ARTIFICIAL INTELLIGENCE

From theory-only to achieving practical purposes, artificial Intelligence is one of the booming technologies nowadays. With plenty of computer science knowledge involved, AI algorithms are applied in various strong-feature applications and applets which are currently supplied in many fields of human life, such as multiple-language translation, virtual personal assistant, smart security system with face recognition, self-driving cars and weather prediction. As the meaning of “Artificial Intelligence” is human-made machines with abilities of working as a human, to accomplish the level of “Intelligence “, programs with specific algorithm are required to run in rational thinking ways like human. (Advani 2020.)

Artificial Intelligence is a popular topic in 21st century, it is one of the advanced research fields of computer sciences. The work of AI began formally soon after World War II, and the word ”Artificial Intelligence” was put forward by Stanford professor John McCarthy in 1956 on Dartmouth workshop. The area of AI research goes further still, it aims not only exists in the theories but also to create entities, which includes a significant variety of subfield, scaling from universal (studying and feeling) to particular, such as playing chess, diagnosing diseases, writing poetry, driving vehicles and proof of mathematical theorems. With the technologies of AI evolving swiftly, its capabilities are been applied in all aspects of society. (Russell & Norvig 2010.)

The utilizations of AI are practically unlimited, and the computer science field is rising. By 2023, the global revenue of the AI market is expected to reach 97.9 billion US dollars. Companies, especially those from the software and information technology services industries, are funding significantly in AI. At the same time, AI-concentrated startups have gained the attention and inclination from investors. From 2015 to 2018, the funding of AI startups has increased almost five times. Machine learning, a type of AI, allows computers to learn without human intervention. In the end, at least 31.7 billion US dollars was invested in this AI category. Many startups are also putting their money into the natural language processing market, which involves speech and speech recognition and text prediction. By 2020, this specific AI market is expected to grow to 12.4 billion dollars. (Khakurel & Penzenstadler & Porras & Knutas & Zhang 2018.)

The main goals of AI consist of deduction and reasoning, knowledge expression, planning, natural language processing, learning, consciousness, and the ability of controlling and shifting objects. Perennial purposes of AI study involve implement creativity, social Intelligence, and general Intelligence. AI has deeply affected every sectors of the society gradually. As Ray Kurzweil said, “Many thousands of AI applications are deeply embedded in the infrastructure of every industry.” (Kurzweil) John McCarthy, the creator of the term "AI", once said that “as soon as it works, no one calls it AI anymore.” (McCarthy) (Chaudhary 2017.)

Early AI research focused on optimizing search algorithms. This method is very meaningful because it can effectively solve many AI tasks by defining a state space and using search algorithms to define and explore the search tree in this state space. By using heuristics to restrict the search area of these search tree often makes the search program easy to handle. This heuristic is resolved through the using of quality damage to convert the program from the thorny problem into solvable problems. This trade-off choices for less computational complexity rather than the best solution for AI programming has become a standard design pattern. (Watson 2008.)

3.1 The definitions of AI

It is the general knowledge that the word "Intelligent" is for describing human smartness. There are two ways to reflect if a human is "Intelligent", which are thoughts and behaviors. "Artificial Intelligence" literally means the artifacts that could be the same intelligent as humans or even surpass human brain. However, the structure of human brain is extreme complicated, and the emotions of human is diversified, it is quite difficult for an artifact to fully imitate human thoughts and actions. As a matter of fact, an artifact that is able to achieve this condition does not exist yet. Nevertheless, past research on AI implemented several part of AI functions to some extent. (Russell, Norvig 2010.)

It is difficult to title machines of the real life as "AI" if following the strict definitions of AI. However, there are three types that could be practically considered as AI, because they reach the essential area of AI definition: first, taking simple machine actions that programmed by developers in advance. For instance, speech or picture recognition systems which can only recognize speech or pictures, and industrial robots in the production line that process the single work repeatedly. Second, seeking answers and solving problems through the rules set by humans. The common example is cleaning robots which can collect data of environment by bumping into obstacles, so that they could build the virtual model of the room in order to correct the route of movement; third, offering the predictable results by measuring the regularity from collected data. For example, the recommendations system on the smart phone which based on users' browsing and purchasing history. Although the computer science researchers have different opinion about definitions of AI, most of explanation could be categorized into four parts: systems that thinking like humans; systems that acting like human; systems that thinking rationally; systems that acting rationally. Figure 5 indicates these four types of AI definition. (Russell, Norvig 2010.)

Thinking Humanly

“The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense.” (Haugeland, 1985)

“[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . .” (Bellman, 1978)

Acting Humanly

“The art of creating machines that perform functions that require intelligence when performed by people.” (Kurzweil, 1990)

“The study of how to make computers do things at which, at the moment, people are better.” (Rich and Knight, 1991)

Thinking Rationally

“The study of mental faculties through the use of computational models.”

(Charniak and McDermott, 1985)

“The study of the computations that make it possible to perceive, reason, and act.”

(Winston, 1992)

Acting Rationally

“Computational Intelligence is the study of the design of intelligent agents.” (Poole *et al.*, 1998)

“AI . . . is concerned with intelligent behavior in artifacts.” (Nilsson, 1998)

Figure 4. Four categories are organized for the definitions of AI (Source: Russell, S. Norvig, P. Artificial Intelligence A Modern Approach Third Edition, Chapter 1, figure 1.1)

3.2 The categories of AI

From the theoretical aspect, the category of AI is divided into three terms as either Artificial Narrow Intelligence (ANI, also known as weak AI) or Artificial General Intelligence (AGI, marked as strong AI) or Artificial Super Intelligence (ASI) based on the level that a machine could reach. (Lawtomated 2019.)

The modern technology has allowed humans to achieve the ANI level in multiple ways such cars with self-driving systems, Google search and email spam filters. However, this kind of machine could only perform according to the algorithm input by developers. And in this level, machines are programmed for the specific tasks and only work in single sphere. In the level of AGI, machines are able to improve and upgrade the knowledge inventory through initiative learning. This kind of behavioral pattern is similar to the human brain, and it could be exploited in various fields. AlphaGo is a significant example of AGI. ASI achieves the most powerful and complicated level of AI, it has ability of surpassing human intelligence in whole area that developed by human, including scientific research, general wisdom and social skills. Figure 6 shows the development of AI which the autonomous driving is in the ANI level. AI level begins from narrow intelligence, then steps into general intelligence and final stage is super intelligence. (Panchal 2018.)

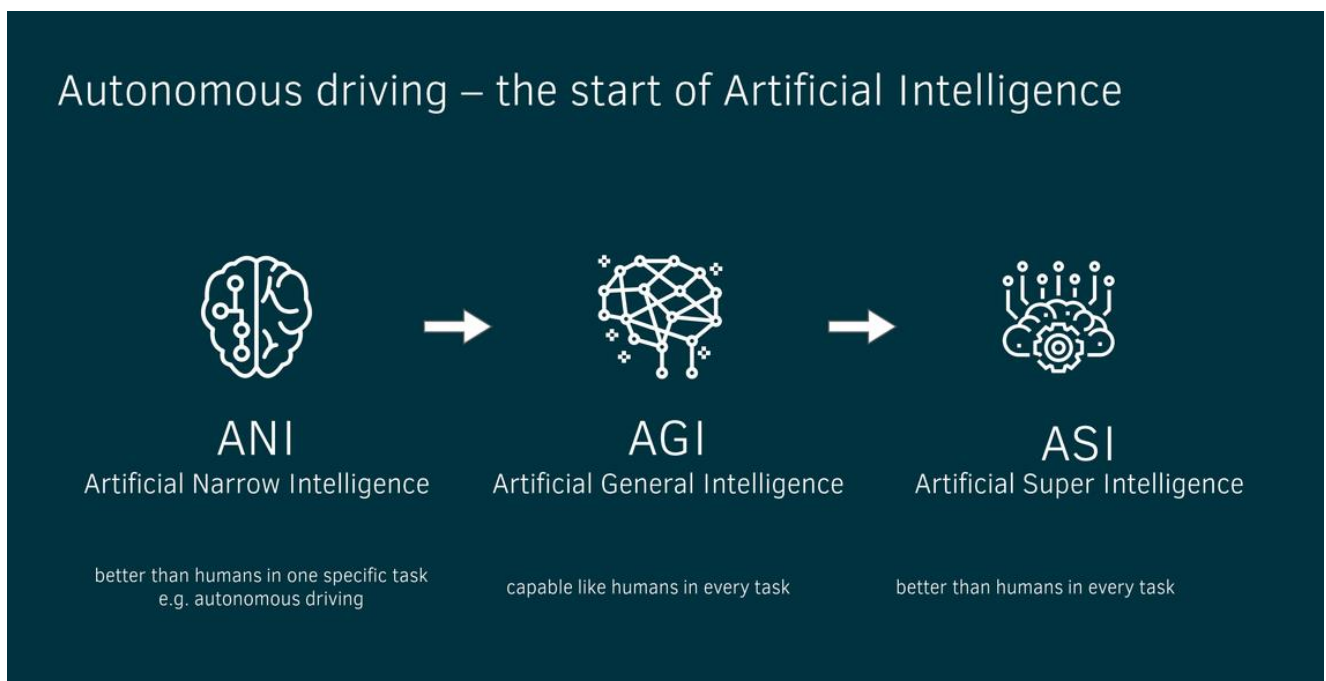


Figure 5. Autonomous driving as the start of Artificial Intelligence level (Source: <https://accilium.com/en/autonomous-driving-one-step-closer-to-artificial-intelligence/>)

3.2.1 Machine Learning

Machine Learning is a subset of artificial intelligence which purpose on enabling machines to work accurately by exploiting intelligent software. The term was introduced by American computer scientist Arthur Samuel in 1959, who also came up with the definition of ML in 1995 that “Machine Learning relates with the study, design and development of the algorithms that give computers the capability to learn without being explicitly programmed.” (Samuel) ML is an interdisciplinarity of computer science and math statistics. Machine Learning algorithms automatically analyze and obtain rules from the training data and predict unknown data. Computer systems with machine learning algorithms could construct mathematical model to complete specific tasks. (Sugomori 2016.)

Three methods are applied for achieving ML, which are supervised learning, unsupervised learning and reinforcement learning. In supervised learning, system gets a set of data samples “X” as inputs, takes “Y” as their outputs. The machine with supervised learning is focus on finding the mapping function as “f” that could convert inputs to outputs. Spam filter is an example of supervised learning. The users marked email (matrix X) as normal or spam (vector Y). The system makes the decision model through learning algorithm for distinguish emails as normal or spam. In unsupervised learning, developers do not have to label the data. Learning algorithm automatically assort similar data inputs into clusters and find concealed rules in the data. This kind of learning algorithm does not predict the outputs. One of the most common examples is recommendation systems with machine learning algorithm could find the same sort of products that purchased by customer. Reinforcement learning is a kind of learning type that enables agents (machines) to learn through an incentive mechanism of rewards or punishments, which is regarded as a series of reinforcements. For instance, fewer bonus of the game makes player agent aware that there was one or more mistakes it made during the game. The agent decides which steps of the movements prior to the reinforcement are most responsible to them. Figure 7 below shows the differences between three methods of machine learning. (Mohammed & Khan & Bashier 2016.)

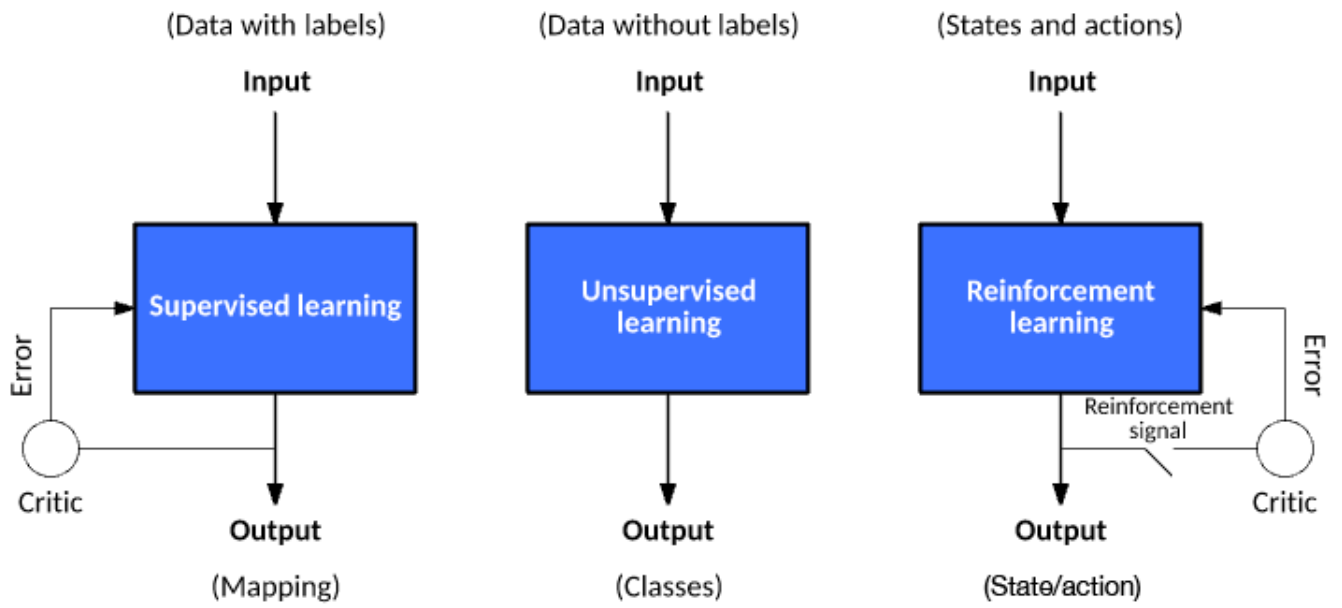


Figure 6. Three learning methods of machine learning algorithms (Source: <https://developer.ibm.com/articles/cc-models-machine-learning/>)

3.2.2 Deep Learning

Deep Learning is a method of machine learning which is able to exploit supervised learning and unsupervised learning simultaneously or separately. It promoted to solve complex problems of certain field in AI such as vision and sound processing. Deep learning is according to representation learning branch of machine learning theory. The concept of representation learning started from the research about artificial neural network, which is a mathematical model of an algorithm that imitates the behavioral feature of animal neural networks and performs distributed parallel information processing. (Wehle 2017.)

High-class and complex abstractions, for instance images or voice, are collected as data representations by the hierarchical learning process. There are multiple hidden layers of models built in artificial neural networks, gathered data was processed level by level, the output was sent to next layer after previous layer finished processing the input. The advantage of a deep learning models is that the speed of processing is higher than general machine learning methods. Deep learning models could recognize the significant features automatically, instead of demanding manual selection for the related features through developers. (Wehle 2017.)

Designing neural networks in any programming language requires understanding the structure and function of artificial neural networks. Traditional algorithmic methods need to execute a set of steps to achieve a defined goal. Artificial neural networks can learn how to solve certain tasks on their own due to their highly interconnected network structure. Artificial neurons have a structure similar to human brain neurons. Natural neurons are composed of nuclei, dendrites and axons. Axons extend themselves to several branches and form synapses with dendrites of other neurons. In addition, the connections between neurons have associated weights that can modify the signal, thereby affecting the output of the neurons. Since the weights are inside the neural network and affect its output, they can be regarded as the internal knowledge of the neural network. Adjusting the weights representing the connections between neurons and other neurons or the external world will reflect the function of the neural network. (Davis 2017.)

3.3 Machine Learning tools and libraries in Java

Machine learning is currently one of the popular technologies. Companies are actively recruiting skilled programmers to fill the gaps in machine learning and deep learning code writing. According to the relevant recruitment statistics, the Python language has now surpassed Java as the most urgently needed machine learning programming skills for employers. But in fact, Java still plays an irreplaceable role in project development, and many popular machine learning frameworks are also written in Java. There are various Java-based open source libraries available for implementing machine learning algorithms. (TIOBE Index 2020.)

Weka integrates machine learning algorithms for data mining. These algorithms could be directly applied to a dataset and utilized in the source code. Weka includes a series of tools, such as data preprocessing, classification, regression, clustering, association rules and visualization. MOA, which stands for Massive Online Analysis, is a popular open source framework for data stream mining, with a very active growing community. It includes a series of machine learning algorithms such as anomaly detection, concept drift detection, and recommendation systems and evaluation tools. Java Machine Learning Library is a series of related implementations of machine learning algorithms. These algorithms, both source code and documentation, are well written. Its main language is Java. Deeplearning4j is the first commercial-grade, open-source, distributed deep learning library written in Java and Scala. However, it is designed to be used in a business environment, not as a research tool. Mallet is a Java-based machine learning toolkit for text files. Mallet supports classification algorithms such as maximum entropy, naive bayes and decision tree classification. H2O is a machine learning API for smart applications. It scales statistics, machine learning and mathematics on big data. H2O is extensible, developers can use simple mathematical knowledge in the core part. (Baeldung 2020.)

4 WEKA

Weka stands for the Waikato Environment for Knowledge Analysis. It is a workbench software written in Java, developed at University of Waikato in New Zealand that could be run at the most of operating systems such as Linux, Windows and Mac. Weka contains a large scope of data preprocessing tools which help access users through a common interface so that they can contrast diverse methods and find out the most appropriate one fast. It also provides implementations of machine learning algorithms which could be exploited in various datasets. By preprocessing a dataset, feeding it into a learning scheme, the resulting classifier and its performance could be analyzed by user without coding. (Frank & Hall & Witten 2016.)

There are three main ways to exploit Weka. The first is to apply a learning scheme to a certain dataset, and then analyze its output to learn more about these data. The second is to use the learned model to predict new instances. The third is to use a variety of learners, and then choose one of them to make predictions based on its performance. The user selects a learning method using the interactive interface menu. Most learning programs have adjustable parameters. The user can modify the parameters through the attribute list or object editor, and then evaluate the performance of the learning scheme through the same evaluation module. Figure 8 is the interface of Weka GUI chooser. According to different applications, the object of data mining can be a variety of data. These data can be various forms of storage, such as databases, data warehouses, data files, streaming data, multimedia and web pages. It can be stored centrally in the data repository or distributed on network servers around the world. (Frank & Hall & Witten 2016.)

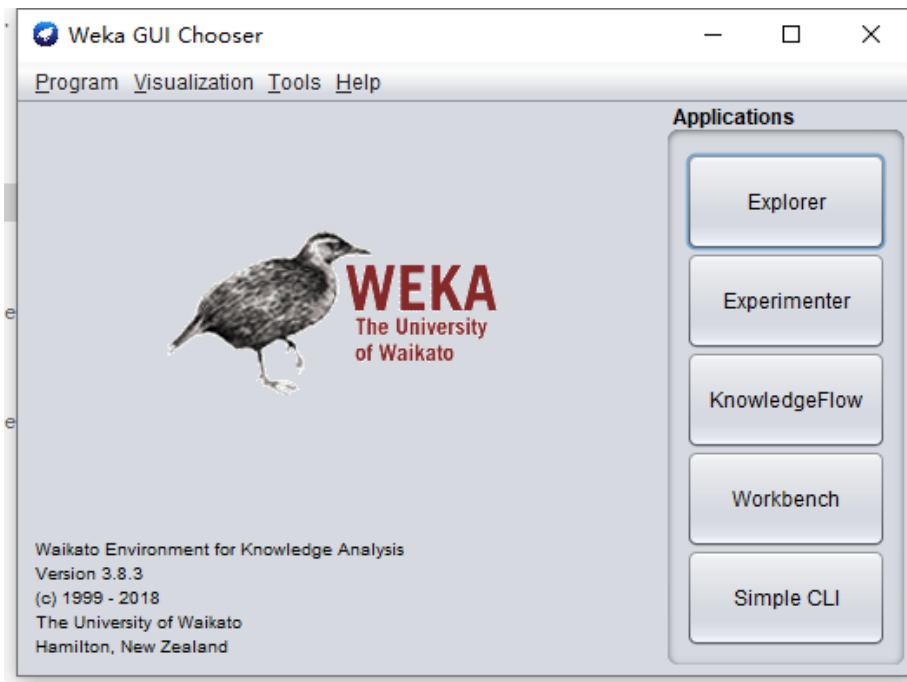


Figure 7. Weka GUI Chooser (Version 3.8.3)

Most datasets exist in the form of database tables and data files. Weka supports reading database tables and data files in multiple formats. Among them, the most used is a file called ARFF format. The ARFF format is a Weka-specific file format. Weka's official document states that ARFF stands for Attribute-Relation File Format. This file is an ASCII text file that describes a list of instances that share a set of attribute structures. It consists of independent and unordered instances and it is the standard method of Weka to represent data collection. ARRF does not involve the relationship between instances. (Frank & Hall & Witten 2016.)

However, there are two shortcomings of Weka compared with Python. First, Weka's pre-processing and result output are more difficult. Although it is convenient for beginners to process data with a little filter, it is easier to write programs like Python when processing large amounts of data. Similarly, although the results can be run out by pressing "Start" in the classification, it is more troublesome for Weka to make the results lead to the format or the next application. Second, the Python package is booming. Although Weka also has a lot of packages, but a closer look will reveal that most of them are old and have not been updated. The Weka suite written in Java is also difficult to rewrite and compile. In contrast, the development of Python is flourishing. Most of late research could be repackaged into Python packages for people to download and use, and there are countless developers studying Python. In this regard, Weka is inferior to Python at all. (MNIST digits Classification with Weka 2017.)

4.1 The explorer experiment

The first step is preparing the training data. Weka application and datasets-UCI.jar (a JAR file including 37 classification problems, originally obtained from the University of California Irvine Machine Learning Repository) are downloaded in the computer. The JAR file was unzipped to folder datasets-UCI, each dataset is stored as ARFF format. Figure 9 shows the dataset files that datasets-UCI.jar contains.

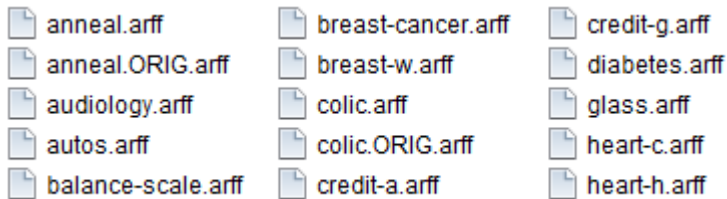


Figure 8. Problems of ARFF format in UCI dataset

The second step is loading data into explorer. Clicked "Explorer" button in the GUI chooser, Opened folder datasets-UCI repository and chose "diabetes.arff" as experimental dataset. Figure 10 displayed the Weka explorer interface reading in the data from the dataset. At the blanket of "Current relation", 768 examples or instances were demonstrated and 9 attributes or features are shown. The best attribute to start with is class, or the label users want to predict. And usually in Weka, that is the last attribute in a dataset.

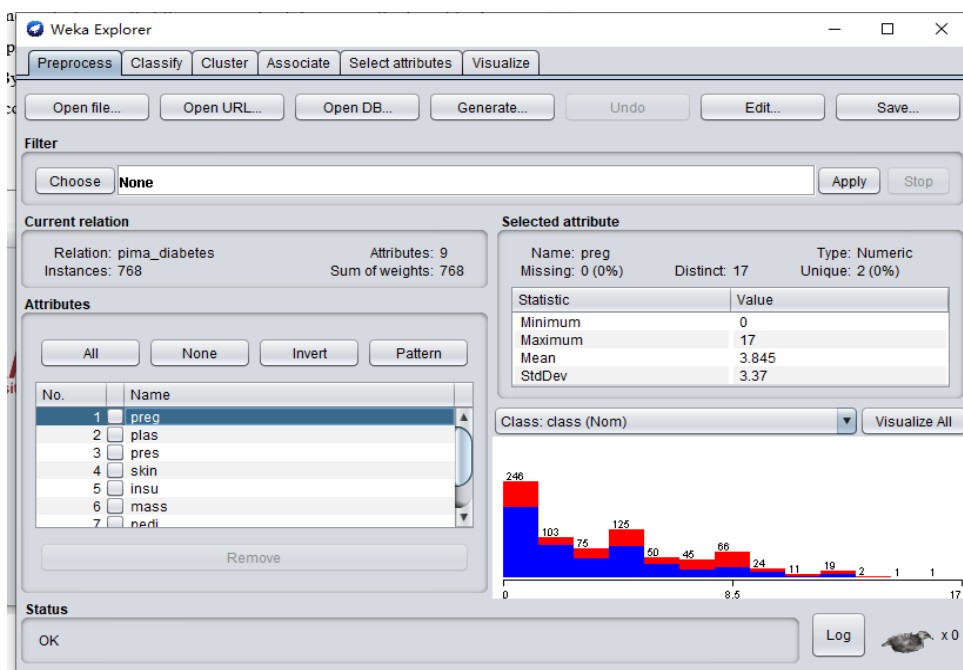


Figure 9. Reading in the diabetes data

The third step is building the decision tree. After headed to the “Classify” tab, author chose the classifier J48, which is an implemented algorithm in Weka. Then author clicked “Start” button and began to invoke, then a C4.5 decision tree was generated. Figure 11 is the output of the classification. J48 is one type of tree that does pruning. For training a linear classifier like logistic regression, under “Functions” and “Logistic” options, author hit to start it. It is more precise to flip back and forth between the two classifiers to compare the results.

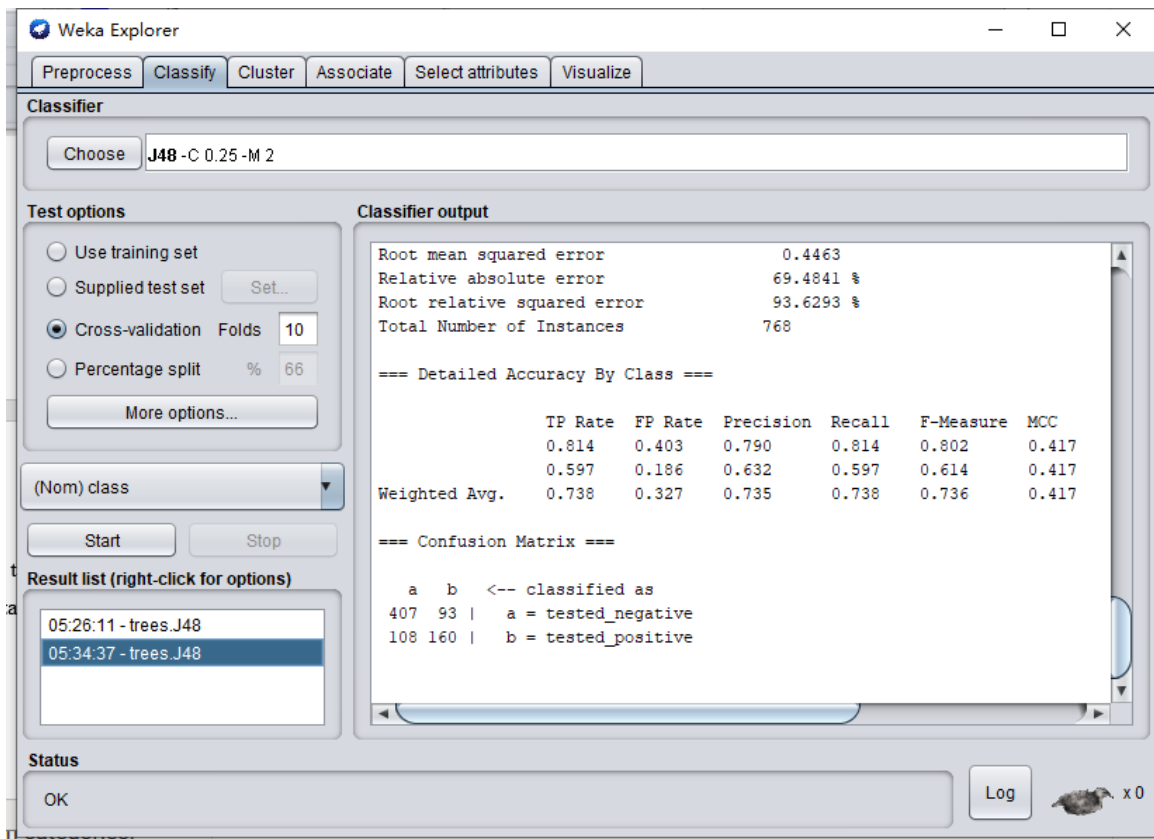


Figure 10. Classifier output of diabetes dataset

In the second data mining experiment, image classification was the main goal. The imageFilter package that downloaded using the Package Manager of Weka was demonstrated. What the imageFilter package does is enable users convert images into features so that they could run image classification experiments for face recognition, scene recognition and object detection. The first step was to get the testing dataset of 2 different animal types of image in one directory, the rhino and the brontosaurus were selected and 43 pictures for each. The content of testing dataset is shown in figure 12.

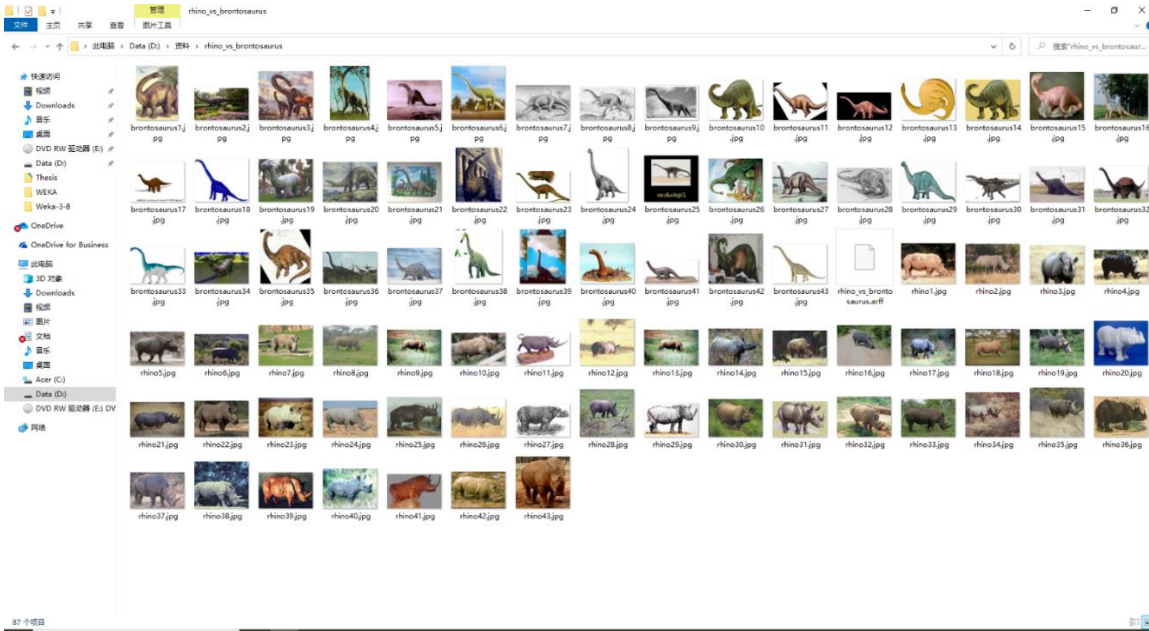


Figure 11. Images of Rhino and Brontosaurus

For the ARFF file, which was applied to the dataset, the first attribute is the string, which has to contain the filenames of the images, and the second attribute contains the class. After the filters was applied, all the filters added these further attributes. Figure 13 shows preprocess interface of the data. At the blanket of “Current relation”, there showed 86 examples or instances, and 2 attributes or features. In the histogram of class, rhino and brontosaurus has the same instance count which is 43.

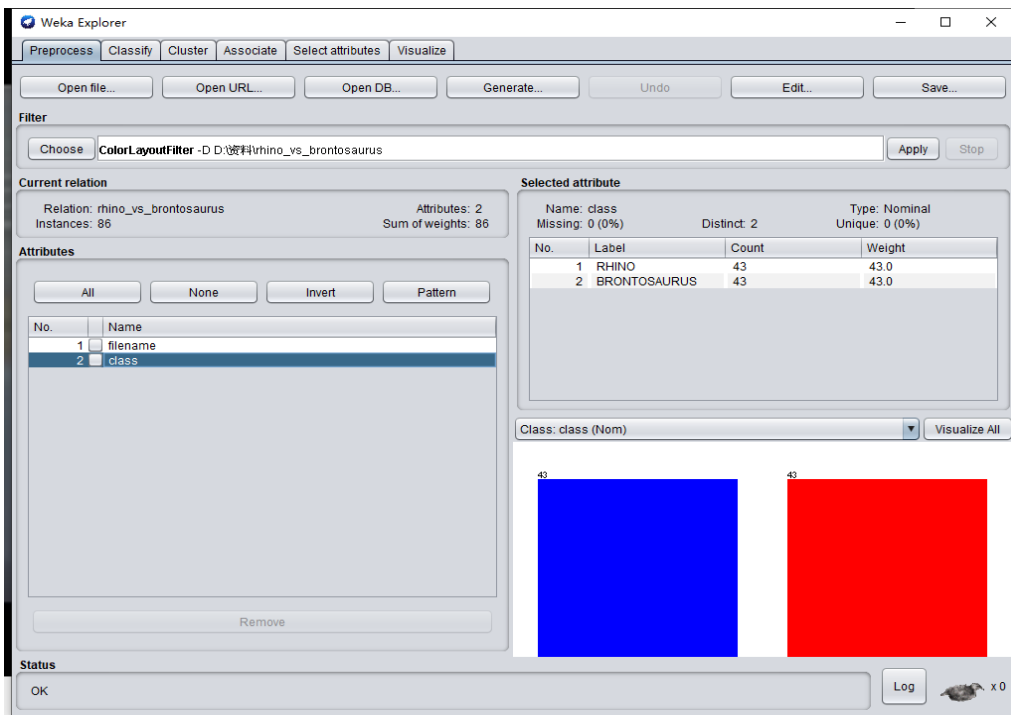


Figure 12. Equal frequency, 43 examples of each in the dataset

All the filters were checked and available under Unsupervised/Instance/imageFilter. The filter which was selected is the ColorLayoutFilter. After inputting the directories that contains all the images, the filter could be applied and it could process every image. Weka read in all images and extracted each feature. After that, additional numeric features were added to the dataset which is the figure 14 showed below.

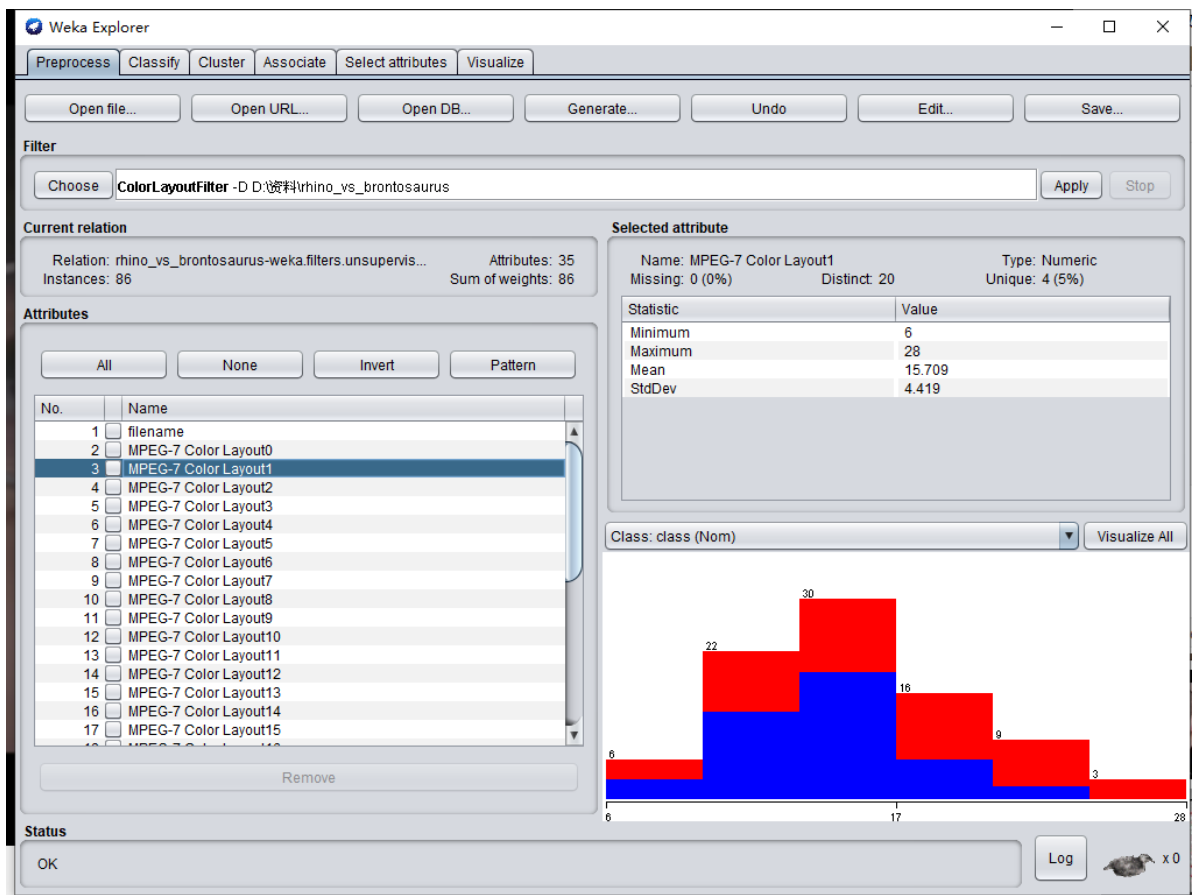


Figure 13. Additional features were added to the dataset

In order to run a classification experiment after running the filter for the first time, the filename attribute should be removed, because it is a string and it could cause problems for different classifiers. After finishing that step, switching to the classify tab then the J48 classifier was used as the testing classifier for classification. The classification result was accuracy with 70.9302% of instances were classified as figure 15 shows. It is possible to apply more filters to the images to improve or decrease the accuracy of classification. For completing that, repeatedly applying different filters to the dataset is needed.

The screenshot shows the Weka Explorer interface with the Classifier tab selected. The classifier chosen is J48-C 0.25-M 2. The test options are set to Cross-validation with 10 folds. The classifier output window displays the following information:

Size of the tree : 17
Time taken to build model: 0.32 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	61	70.9302 %
Incorrectly Classified Instances	25	29.0698 %
Kappa statistic	0.4186	
Mean absolute error	0.2956	
Root mean squared error	0.5242	
Relative absolute error	59.067 %	
Root relative squared error	104.7432 %	
Total Number of Instances	86	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.721	0.302	0.705	0.721	0.713	0.419	0.725	0.667	RHINO
	0.698	0.279	0.714	0.698	0.706	0.419	0.725	0.680	BRONTOSAURUS
Weighted Avg.	0.709	0.291	0.709	0.709	0.709	0.419	0.725	0.673	

=== Confusion Matrix ===

```

a b <-- classified as
31 12 | a = RHINO
13 30 | b = BRONTOSAURUS

```

The result list shows a single entry: 20:42:27 - trees.J48. The status bar at the bottom indicates 'OK' and has a 'Log' button.

Figure 14. Classifier output

4.2 The class structure Weka

Weka is mainly written in Java programming language. In object-oriented programming, a Java program is implemented as a class, which consists of variables and methods operating on them, all of them together define the behavior of an object included in the class. The instantiation of class with assigned values to class variables is called object, which is also named as an instance of the class. The implementation of a particular machine learning algorithm is represented by a class. For example, the functionality of J48 class is building a decision tree. When the JVM executes the J48 class, it generates an instance of the J48 through allocating memory for establishing and storing a decision tree classifier. Large-size Java programs are divided into several classes that organized into directories which calls packages. Packages are formed in a hierarchy. The classifiers package belongs to the overall Weka package, and it contains the J48 package as its sub-package. Figure 16 illustrates the interface of Weka package manager. (Frank & Witten 2000.)

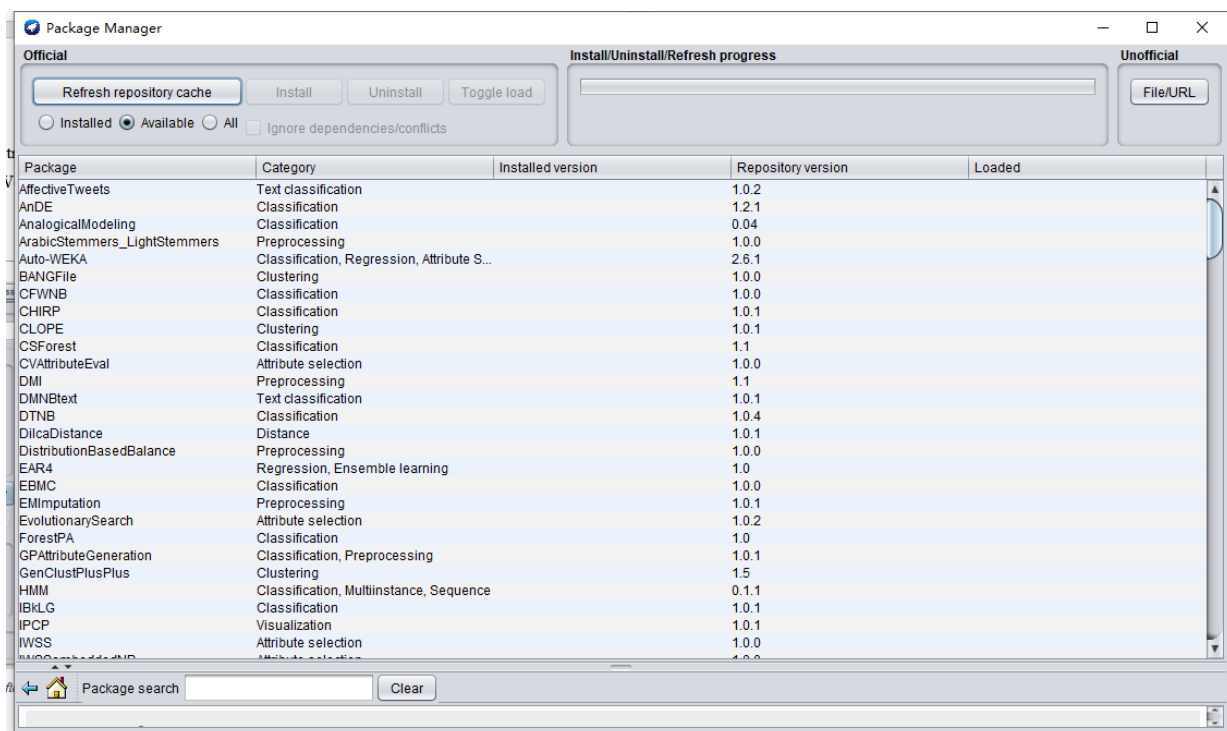


Figure 16. Package Manager in Weka

4.2.1 The Weka.core package

The weka.core package forms the kernel of Weka system. The classes which it contains could be almost accessed to each other. The weka.core includes a kind of class called “interface”, however, interface does not perform any operation, it only has methods without actual implementation. Rest of classes provide code to the methods of a specific interface for implementation. For example, Copyable interface defines methods that are implemented by classes that can produce “shallow” copies of their objects.

Attribute, Instance and Instances are the fundamental classes in the weka.core package. An object of Attribute class is represented as an attribute, which essentially includes name, type and value in special cases. An object of Instance class has the attribute values of a particular instance. An object in Instances class contains a sequence of instances. Figure 17 shows the method of Copyable interface. (Frank & Witten 2000.)

Interface weka.core.Copyable

public interface **Copyable**

Interface implemented by classes that can produce "shallow" copies of their objects. (As opposed to clone(), which is supposed to produce a "deep" copy.)

Version:

\$Revision: 1.2 \$

Author:

Eibe Frank (eibe@cs.waikato.ac.nz)

Method Index

- **copy()**

This method produces a shallow copy of an object.

Methods

- **copy**

public abstract Object copy()

This method produces a shallow copy of an object. It does the same as the clone() method in Object, which also produces a shallow copy.

Figure 17. Method of Copyable interface (Source: <https://www.dbs.ifi.lmu.de/~zimek/diplomathesis/implementations/EHNDs/doc/weka/core/Copyable.html/>)

4.2.2 The Weka.classifiers package

The weka.classifiers package involves significant Classifier class, which defines overall architecture of any scheme for classification or numeric prediction. Classifier class contains methods buildClassifier() and classifyInstance(). In object-oriented programming, each learning algorithm is represented by a subclass of Classifier class, which inherits and implements those two methods mentioned before. Each scheme redefines buildClassifier() and classifyInstance() respectively according to how it builds a classifier and how it classifies instances. This provides a generic interface for constructing and exploiting classifiers from other Java programming part.

DistributionClassifier is a sub-class of Classifier class, it defines the method distributionForInstance(), which returns a probability distribution of a certain instance. Each classifier which has function of computing class probabilities belongs to the DistributionClassifier class and implementing the method distributionForInstance(). All of these classes above belong to java.lang.Object class, the structure helps for building a tree relevant to the hierarchy of Weka classes. Figure 18 illustrates the usage of method distributionForInstance(): the parameter is the instance to be classified and it returns an array. (Frank & Witten 2000.)

Methods

• distributionForInstance

```
public abstract double[] distributionForInstance(Instance instance) throws Exception
```

Predicts the class memberships for a given instance. If an instance is unclassified, the returned array elements must be all zero. If the class is numeric, the array must consist of only one element, which contains the predicted value.

Parameters:

instance - the instance to be classified

Returns:

an array containing the estimated membership probabilities of the test instance in each class (this should sum to at most 1)

Throws: Exception

if distribution could not be computed successfully

Figure 18. Method distributionForInstance() (Source: <https://weka.sourceforge.io/doc.dev/weka/classifiers/Classifier.html#distributionForInstance-weka.core.Instance-/>)

5 PROJECT EXPLANATION

Naive Bayes classifier is a classic machine learning algorithm which is based on Bayes theorem and independent assumption of feature. According to probability theory and Bayes theory, the classifier is able to predict the category of a group of data set sample. It is an algorithm which could be easily implemented to process and classify the training data fast. The most typical application of the Naive Bayes classifier is the identification and filtering of spam, which is the function that the project had. In the case of a very large amount of data, the accuracy of the identification can be close to 100%, and the implementation idea is not complicated. (Kaluza 2016.)

5.1 Naive Bayesian algorithmic reasoning

The origin of the word "naive" in Naive Bayes is to assume that the features are independent of each other. The assumption makes the Naive Bayes algorithm simple, but it frequently sacrifices certain classification accuracy. The Bayesian formula is Formula 1. Assume that the two types of mail are two random events A and B, where A is spam and B is ham mail, which are all random events in the sample space S of all self-learning E. T is a collection of mail words, where T_i is an element of T. (Bayesian Multivariate Statistical Inference Theory 2006.)

Formula 1

$$P(B_i | A) = \frac{P(A | B_i)P(B)}{\sum_{j=1}^n P(A | B_j)P(B_j)}$$

Formula 2

$$P(A | T_i) = \frac{P(T_i | A) * P(A)}{P(T_i | A) * P(A) + P(T_i | B) * P(B)}$$

According to the Formula 2, the amount of spam in the sample space is equivalent to ham mail, so $P(A) = P(B) = 0.5$, assume that $f_A(T_i) = P(T_i | A)$, $f_B(T_i) = P(T_i | B)$, Formula 3 is for calculating the probability of whether a mail is spam in the presence of a word. After the keyword list of rejection is learned, in the case of multi-keyword filtering, the combining probability formula can be obtained as Formula 4. From Formula 4, the probability of spam can be calculated when the email is received, and then determine whether it is spam according to the set threshold. (Bayesian Multivariate Statistical Inference Theory 2006.)

Formula 3

$$P(A | T_i) = \frac{f_A(T_i)}{f_A(T_i) + f_B(T_i)}$$

Formula 4

$$P(A | T_1, \dots, T_n) = \frac{P(A | T_1) * P(A | T_2) * \dots * P(A | T_n)}{[P(A | T_1) * P(A | T_2) * \dots * P(A | T_n)] + [(1 - P(A | T_n)) * (1 - P(A | T_{n-1})) * \dots * (1 - P(A | T_1))]}$$

The steps of the algorithm: First, process the training data and propose each training data and its corresponding label. Second, generate a vocabulary based on the training data. Third, vectorize each training sample, then combine the generated vocabulary to train a Naive Bayes classifier. Fourth, process the test data and propose each test data and its corresponding label. Fifth, vectorize each test data and classify it using Naive Bayes classifier. Sixth, compare the probability of spam and ham to determine which category it belongs to. (Bayesian Multivariate Statistical Inference Theory 2006.)

5.2 Selection of datasets

The implementation of Naive Bayesian algorithm is not complicated. The main difficulty lies in the processing of the dataset, which improves the accuracy of the algorithm. Therefore, selecting suitable training dataset and testing dataset is required before setting classifier. In the project, there is an ARFF data file "TrainingData.arff" which contains total of 5000 messages with spam or ham label was used as the training dataset, the testing dataset "TestingData.arff" is an ARFF data file that contains 525 messages with spam or ham label, also, there are two txt files as raw datasets respectively as the figure 19 shows below.





 test.txt	2020/3/17 22:44	文本文档	44 KB
 TestingData.arff	2020/3/11 23:08	ARFF Data File	45 KB
 train.txt	2020/3/17 22:45	文本文档	420 KB
 TrainingData.arff	2020/3/24 21:22	ARFF Data File	437 KB

Figure 19. Training dataset and testing dataset

Figure 20 below indicates the curt content of training dataset. In the file TrainingData.arff. "@relation 'SMS spam'" defined relation name, which is string data type. In case of the name contains spaces, it is required to be quoted. "@attribute label {spam, ham}" defined attribute "label" which includes two selections with "spam" and "ham". "@attribute text string" defined the data type of attribute "text" is string. The lines that start after "@data" are the instance data in the dataset. Each line represents an instance, the attribute values are arranged in the order of the attributes and separated by commas, and the carriage return indicates the end of the instance. The format is "label,'text'".

```

TrainingData.arff - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
@relation 'SMS spam'

@attribute label {spam,ham}
@attribute text string

@data
ham,'Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...'
ham,'Ok lar... Joking wif u oni...'
spam,'Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over1
ham,'U dun say so early hor... U c already then say...'
ham,'Nah I don't think he goes to usf, he lives around here though'
spam,'FreeMsg Hey there darling it's been 3 week's now and no word back! I\'d like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv'
ham,'Even my brother is not like to speak with me. They treat me like aids patent.'
ham,'As per your request \'Melle Melle (Oru Minnaminunginte Nurungu Vettam)\' has been set as your callertune for all Callers. Press *9 to copy your friends Ca
spam,'WINNER!! As a valued network customer you have been selected to receive a £900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours o
spam,'Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 0800298603
ham,'I\'m gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I\'ve cried enough today.'
spam,'SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info'
spam,'URGENT! You have won a 1 week FREE membership in our £100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW
ham,'I\'ve been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have
ham,'I HAVE A DATE ON SUNDAY WITH WILL!!'
spam,'XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap.xxxmobilemovieclub.com?n=QJKGIGHJIGCBL'
ham,'Oh k...i\'m watching here:)'
ham,'Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.'
ham,'Fine if thats the way u feel. Thats the way its gota b'
spam,'England v Macedonia - dont miss the goals/team news. Txt ur national team to 87077 eg ENGLAND to 87077 Try:WALES, SCOTLAND 4txt/ú1.20 POBOXox36504W45WQ 1
ham,'Is that seriously how you spell his name?'
ham,'Ifs going to try for 2 months bc he only taking'

```

第 1 行, 第 1 列 100% Unix (LF) UTF-8

Figure 20. TrainingData.arff opened as txt file

5.3 Importing Weka Packages

NetBeans IDE 8.2 was the development environment for the project. After creating the new application named “Spam_Filter”, for exposing the weka API to NetBeans, so that weka packages could be imported into java source code, it is necessary to add an archive file weka.jar into libraries of the project as the figure 21 shows below.

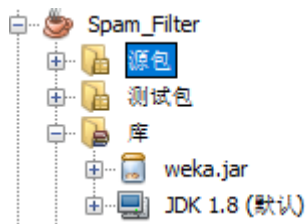


Figure 21. Libraries included weka.jar and JDK 1.8

Weka packages were available after weka.jar file was included in the libraries. First, focus on Attribute, and Instances in the core classes under the weka.core package. The collection of classifying samples is saved as a global variable in the form of Instances, where each sample is an Instance. Each classifying sample contains numbers of Attribute, and one Attribute of each Instance corresponds to a variable value. In other words, the sample set Instances is a large table, the column direction is Instance, the row direction is Attribute, and the table stores the value of an Instance corresponding to the attribute.

The weka.core.converters package is used to find out the file sources of training dataset and testing dataset that in arff format, and then read data from those files either in incremental or batch mode. The weka.core.tokenizers.NgramTokenizer includes the method that splits a string into an n-gram with min and max grams, used to add the tokenizer that set min and max word number to the filter. Figure 22 shows weka.core packages included in the source code. The weka.classifiers.Evaluation class was exploited to evaluate machine learning models. The weka.classifier.meta.FilteredClassifier class arbitrarily classified the data that has been passed through a random filter. For constructing and implanting the multinomial Naive Bayesian classifier, weka.classifier.NaiveBayesMultinomial was applied.

```

import weka.core.Instances;
import weka.core.Instance;
import weka.core.Attribute;
import weka.core.DenseInstance;
import weka.core.converters.ArffSaver;
import weka.core.converters.ArffLoader.ArffReader;
import weka.core.tokenizers.NGramTokenizer;

```

Figure 22. Weka.core packages used in the source code

The `weka.filters.unsupervised.attribute.StringToWordVector` contains converter that transform string attributes into a set of attributes denoting word occurrence information from the text contained in the strings. In other word, this class convert the string attribute into a set of word attribute, the value of the attribute could be specified in the parameter, such as the 0-1 variable that representing whether the word appears in the instance. It is a class that commonly used in the text mining. Figure 23 below is the `weka.classifier` packages which are imported into the code.

```

import weka.classifiers.Evaluation;
import weka.classifiers.meta.FilteredClassifier;
import weka.classifiers.bayes.NaiveBayesMultinomial;
import weka.filters.unsupervised.attribute.StringToWordVector;

```

Figure 23. Weka.classifiers package and `StringToWordVector` class are imported in the source code

5.4 Training data

After understanding the structure of the data set, it is time to process the data set. First job is to train the data set. Instances "TrainingData" was used for define the training data instances. The data type of defined list `ArrayList<>` is limited to the object "Attribute", for generating new object which included "label" and "text" attributes, the class declared and instantiated the object as "Weka_Attributes". The locations and names of files are defined in the string variable "Training_Data" and "TrainingData_ARFF". Figure 24 below shows the defined instances, attributes of training data and locations of training dataset files

```

45     private Instances TrainingData;
        private ArrayList <Attribute> Weka_Attributes = new ArrayList < > ();
        private static final String Training_Data = "dataset\\train.txt";
        private static final String TrainingData_ARFF = "dataset\\TrainingData.arff";

```

Figure 24. Defined instances, attributes of training data and locations of training dataset files

In the class `WekaClassifier()` created in the project, new array list was built in string data type to add values of attribute "label". The labels of mail were distinguished as "spam" and "ham". Attribute class from `weka.core` package was used to generate two attribute objects of data set. Among them `ClassAttribute` included array list of mail label value "spam" and "ham". Before added the attribute type "text" into object `AttributeText` to save the message, `List<String>` was used to initialize the content list as null. At last, both of these two attribute objects were put in the attribute list `Weka_Attributes`. In this way, the feature vector was declared. Figure 25 illustrates that the attributes "label" and "text" were added and "label" was assigned values.

```

65     ArrayList < String > LabelAttributeValues = new ArrayList < > ();
66     LabelAttributeValues.add("spam");
67     LabelAttributeValues.add("ham");
        Attribute ClassAttribute = new Attribute("label", LabelAttributeValues);
69     Attribute AttributeText = new Attribute("text", (List < String > ) null);
70
71     Weka_Attributes.add(ClassAttribute);
72     Weka_Attributes.add(AttributeText);

```

Figure 25. Attributes "label" and "text" were added and "label" was assigned values

The following step is loading training data and setting feature generators. TRANSFORM() function was built to handle training dataset and initiate the tokenizer. To load the training dataset in space separating text file and convert it to Arff format, the string variable Training_Data for indicating text format training dataset file name and location was added into function loadRawDataset() shown in the figure 26. The function saveArff() saved instances object which holding data into the ARFF format dataset file TrainingData_ARFF.

```

81 | | | | | TrainingData = loadRawDataset(Training_Data);
82 | | | | | saveArff(TrainingData, TrainingData_ARFF);

```

Figure 26. Loading and saving dataset content

Machine learning model algorithms require that the input data must be numeric, so the StringToWordVector class shown in the figure 27 was applied for converting those text in the file into numeric data. By using the setAttributeIndices() class to determine the range of attribute that is to be transformed, since attributes are indexed from 1 and the attribute “text” is in the final, so it would convert on the last attribute.

```

85 | | | | | StringToWordVector filter = new StringToWordVector();
86 | | | | | filter.setAttributeIndices("last");

```

Figure 27. Converting text to feature vector

To implement the tokenizer, the class NGramTokenizer was used to create a tokenizer which divided the string into an n-gram with the minimum and maximum word length as 1. Range was set as any non-word character and the tokens are regulated as lower case. Both conditions were added into the filter. Figure 28 below shows the code that using tokenizer split the string.

```

89 | | | | | NGramTokenizer tokenizer = new NGramTokenizer();
90 | | | | | tokenizer.setNGramMinSize(1);
91 | | | | | tokenizer.setNGramMaxSize(1);
92 | | | | |
93 | | | | |
94 | | | | | tokenizer.setDelimiters("\\W");
95 | | | | | filter.setTokenizer(tokenizer);
96 | | | | | filter.setLowerCaseTokens(true);
97 | | | | | CLASSIFIER.setFilter(filter);

```

Figure 28. Using tokenizer to split the string

The function `fit()` shown in figure 29 was built to implement the classifier with the training data. The class `buildClassifier` classified the text instances in the training dataset. Try-catch statement was used for handling the exception, the warning message would be generated by logger after the exception is caught. This class is for running an arbitrary classifier on data that has been passed through an arbitrary filter. Training data and test instances will be processed by the filter without changing their structure.

```
110 public void fit() {  
111     try {  
112         CLASSIFIER.buildClassifier(TrainingData);  
113     } catch (Exception e) {  
114         lgr.warning(e.getMessage());  
115     }  
116 }
```

Figure 29. Building the classifier

5.5 Testing data

Regression testing is usually used to evaluate the accuracy of the classifier. The simplest method is to classify the training data with the constructed classifier, and then give the correct rate evaluation based on the result. But this is not a good method, because the use of training data as detection data may lead to overly optimistic results due to overfitting. So, a better method is to use specialized data for detecting the accuracy of the built classifier, which is called “testing data”. The function `evaluate()` in figure 30 was created to assess the classifier by using testing dataset.

First step is to load the testing data and Instance class from the `weka.core` package was exploited to generate the instance “testData”. If-else statement was used to detect the location of the testing data ARFF file if the file exists. The data from `TestingData_ARFF` was read into the instance “testData”. Then the instance “eval” which is constructed by the Evaluation class was used to evaluate the classifier learning model “CLASSIFIER” by the object of the test data. The result of evaluation would be return as string data type and in case of error there was a message sent by try-catch, then it would be return to the logger “lgr” as the warning.

```

155 public String evaluate() {
156     try {
157         // Load test data
158         Instances testData;
159         if (new File(TestingData_ARFF).exists()) {
160             testData = loadArff(TestingData_ARFF);
161             testData.setClassIndex(0);
162         } else {
163             testData = loadRawDataset(Testing_Data);
164             saveArff(testData, TestingData_ARFF);
165         }
166
167         Evaluation eval = new Evaluation(testData);
168         eval.evaluateModel(CLASSIFIER, testData);
169         return eval.toSummaryString();
170     } catch (Exception e) {
171         lgr.warning(e.getMessage());
172         return null;
173     }
174 }

```

Figure 30. Evaluating the classifier model

5.6 Evaluation result

In the end, at the main method of RUN.java, loadModel() function was used to load the learning model which is to be used as classifier. After the new object “wt” was generated, if-else statement was used to launch functions for extracting training datasets, creating the filter then implementing the classifier by the training data. The learning model was saved by using function saveModel(). Two testing messages were written into the logger and processed to prediction. The prediction result was printed by using evaluate() function. Figure 31 below shows the code of main() function.

```
public class RUN extends WekaClassifier {
    /**
     * Main method. With an example usage of this class.
     */
    public static void main(String[] args) throws Exception {
        final String MODEL = "models";

        WekaClassifier wt = new WekaClassifier();

        if (new File(MODEL).exists()) {
            wt.loadModel(MODEL);
        } else {
            wt.TRANSFORM();
            wt.fit();
            wt.saveModel(MODEL);
        }

        // Run few predictions
        Jgr.info("I think we could meet the day after tomorrow in the school and talk." + wt.predict("I think we could meet the day after tomorrow in the school and talk."):
        Jgr.info("Congratulations! You have won the 10,000 EUR prize!" + wt.predict("Congruatulation! You have won the 10,000 EUR prize!"));

        // Run evaluation
        Jgr.info("Evaluation Result:\n" + wt.evaluate());
    }
}
```

Figure 31. Main method for running classifier

For building the classifier, after the function TRANSFORM() was launched, the training data was fed to create the filter, then the filter was added into classifier. Two short messages were typed for testing the prediction of the classifier. Because of the key words filtering, the first was labeled as “ham” and the second one was given as “spam”. According to the result, the prediction was consistent of the definition of testing messages. In the evaluation result, there were 522 instances which classified correctly in the testing dataset and the rest 3 messages were not classified properly as shown in figure 32. The summary of classification result came to precision.

```
输出
Spam_Filter (debug) x  调试器控制台 x
debug:
十月 07, 2020 2:28:28 上午 WekaClassifier loadModel
信息: Loaded model: models
十月 07, 2020 2:28:28 上午 RUN main
信息: 'I think we could meet the day after tomorrow in the school and talk.' is ham
十月 07, 2020 2:28:28 上午 RUN main
信息: 'Congratulations! You have won the 10,000 EUR prize!' is spam
十月 07, 2020 2:28:29 上午 RUN main
信息: Evaluation Result:

Correctly Classified Instances      522          99.4286 %
Incorrectly Classified Instances    3            0.5714 %
Kappa statistic                    0.9763
Mean absolute error                 0.011
Root mean squared error             0.0742
Relative absolute error             4.5086 %
Root relative squared error         21.3327 %
Total Number of Instances          525

成功构造 (总时间: 8 秒)
```

Figure 32. Output after running

6 CONCLUSION

Java is still considered a good choice for AI development. AI is related to searching algorithms, artificial neural networks and genetic programming. Java provides these benefits: easy to use, easy to debug, package services, simplifying the work of large projects, graphical representation of data, and better user interaction. This is the reason why Java is used as the programming language of the presented project.

The technology of spam recognition and filter is one of the aspects in AI developing in today's world. Spam and ham mail classification could bring convenience and efficiency to the mailbox users. Although Python is the most popular language for machine learning nowadays, the machine learning algorithm collection written by Java such as Weka could also handle the data classification jobs smoothly, thus, for the purpose of achieving the goal of spam filter, Weka workbench was used in the present study.

By applying Naive Bayes algorithm from Weka workbench software, the data classification is easy to implement. The strong point of the classifier is that the storage only involves two-dimensional and the time and the space cost in the classification process is small. The classifier is "naive" due to the assumption of conditional independence of features. However, even if the algorithm is simple and efficient since the conditional independence assumption would not be true in actual situations. This will lead to the greater the correlation between the features in the actual process, the bigger the classification error.

Normally regression testing is used to evaluate the accuracy of the classifier. The simplest method is to classify the training data with the constructed classifier, and then give the correct rate evaluation based on the result. But this is not a good method, because using training data as detection data may lead to too optimistic results due to overfitting, so a better method which used in the implementation is to divide the training data into two at the beginning of the construction. One is the training dataset, another one is the testing dataset for verifying the accuracy of the classifier. In the experiment, the test dataset composed of 525 messages verifies the classifier well, which helps to prove that the classifier is more reliable adopting test data.

REFERENCES

Daume, H III. 2017. A Course in Machine Learning. Accessed: 15th October

Watson, M. 2008. Practical Artificial Intelligence Programming with Java. Accessed: 15th October

Joost, N. Egbert, J. Walter, A. Putten, P. 2009. Artificial Intelligence: Definition, Trends, Techniques, and Cases. Accessed: 15th October

Wehle, H. 2017. Machine Learning, Deep Learning, and AI: What's the Difference? Available: https://www.researchgate.net/publication/318900216_Machine_Learning_Deep_Learning_and_AI_What%27s_the_Difference/ Accessed: 15th October

Khakurel, J. Penzenstadler, B. Porras, J. Knutas, A. Zhang, W. 2018. The Rise of Artificial Intelligence under the Lens of Sustainability. Available: https://www.researchgate.net/publication/328653902_The_Rise_of_Artificial_Intelligence_under_the_Lens_of_Sustainability/ Accessed: 15th October

Sugomori, Y. 2016. Java Deep Learning Essentials. Available: <https://sites.google.com/site/aduh08kelangan1/POILJo784HDkg1311/> Accessed: 15th October

Witten, H. Hall, A. Frank, E. 2016. The WEKA Workbench. Accessed: 15th October

Kaluza, B. 2016. Machine Learning in Java. Accessed: 15th October

Russell, S. Norvig, P. 2010. Artificial Intelligence A Mordern Approach Third Edition. Accessed: 15th October

Schildt, H. 2011. Java: A Beginner's Guide, Eighth Edition. Available: <https://sites.google.com/site/sijiloro321telu22/aBOt8201wesen2680/> Accessed: 15th October

Advani, V. 2020. What is Artificial Intelligence? How does AI work, Types and Future of it? Available: <https://www.mygreatlearning.com/blog/what-is-artificial-intelligence/> Accessed: 15th October

Nicholson, C. 2020. Java Tools for Deep Learning, Machine Learning and AI. Accessed: 15th October

- Sakovich, N. 2018. Java: Is It the Best Language for Artificial Intelligence? Available: <https://www.sam-solutions.com/blog/java-is-it-the-best-language-for-artificial-intelligence/> Accessed: 15th October
- Austerlitz, H. 2003. Data Acquisition Techniques Using PCs. Accessed: 15th October
- Parahar, M. 2020. Differences between JDK, JRE and JVM. Accessed: 15th October
- TIOBE Index. 2020. Available: <https://www.tiobe.com/tiobe-index/> Accessed: 15th October
- Lawtomated. 2019. AI for Legal: ANI, AGI and ASI. Available: <https://medium.com/@lawtomated/ai-for-legal-ani-agi-and-asi-cea6e7a4079e/> Accessed: 15th October
- Venners, B. 1998. Inside the Java Virtual Machine. Accessed: 15th October
- Chaudhary, S. 2017. Artificial Intelligence 101: How to get started. Available: <https://www.hackerearth.com/blog/developers/artificial-intelligence-101-how-to-get-started/> Accessed: 15th October
- Panchal, S. 2018. Types Of Artificial Intelligence And Examples. Available: <https://medium.com/predict/types-of-artificial-intelligence-and-examples-4f586489c5de/> Accessed: 15th October
- Mohammed, M. Khan, M. Bashier, E. 2016. Machine Learning: Algorithms and Applications. Accessed: 15th October
- Davis, S. 2017. Navigating Neural Networks. Accessed: 15th October
- Baeldung. 2020. Overview of AI Libraries in Java. Available: <https://www.baeldung.com/java-ai/> Accessed: 15th October

SOURCE OF FIGURES

Figure 1: Arora,K Using Java in Artificial Intelligence Programming. Available: <https://www.oodles-technologies.com/blogs/using-java-in-artificial-intelligence-programming/>

Figure 3: Venners,B. Inside the Java Virtual Machine, Chapter 5, figure 5-1

Figure 4: Oracle, JDK 5.0 Documentation. Available: <https://docs.oracle.com/javase/1.5.0/docs/>

Figure 5: Russell,S. Norvig,P. Artificial Intelligence A Mordern Approach Third Edition, Chapter 1, figure 1.1

Figure 6: Kolb,D. Autonomous Driving - One step closer to artificial intelligence, figure 1. Available: <https://accilium.com/en/autonomous-driving-one-step-closer-to-artificial-intelligence/>

Figure 7: Jones,T. Models for Machine Learning, figure 1. Available: <https://developer.ibm.com/articles/cc-models-machine-learning/>

Figure 17: Interface Copyable. Available: <https://www.dbs.ifi.lmu.de/~zimek/diplomathesis/implementations/EHNDs/doc/weka/core/Copyable.html/>

Figure 18: Method Summary, distributionForInstance. <https://weka.sourceforge.io/doc.dev/weka/classifiers/Classifier.html#distributionForInstance-weka.core.Instance-/>