JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

ALGORITHMS & PROBLEM SOLVING PROJECT

ON

"SPAM FILTERING"

(Even Semester 2016)

SPAM ASSASSIN

(A SPAM DETECTION PROGRAM)

Team Members (Batch-B9):

Vaibhav Grover (14102197) Rishabh Devgan (14103279)

Hardik Gulati (14103004) Aatish Bansal (14103285)

Abstract

Spam (junk-email) identification is a well-documented research area. A good spam filter is not only judged by its accuracy in identifying spam, but also by its performance. This project aims to implement a Naive Bayesian spam filter, '*Spam Assassin: A Spam Detection Program*'. In addition,
the project investigates the actual effect of the Porter Stemming algorithm on such filter.

Acknowledgement

We are grateful to **Miss Somya Jain** for her valuable suggestions and assistance with **Porter's Algorithm**. We extend our heartfelt gratitude to her for significantly directing the project at the time of its initialization.

Contents

Chapter 1

Introduction

The rapid growth of Internet has popularised E-mail as an effective means to communicate between people. At the same time, it has encouraged a new form of advertising known as spam or junk-email. Sophisticated spammers forge their e-mail headers so that it can bypass many filters relying on address checking. In this project, the filter relies on the actual message content to distinguish spam emails. There are three distinct processes. First, the filter is supplied with large amount of training data, including both genuine and spam e-mails. Second, the filter removes redundant words and smooths data by applying the Porter Stemming algorithm. Finally, the testing e-mails are passed into the filter for classification.

1.1 Algorithms & approaches used

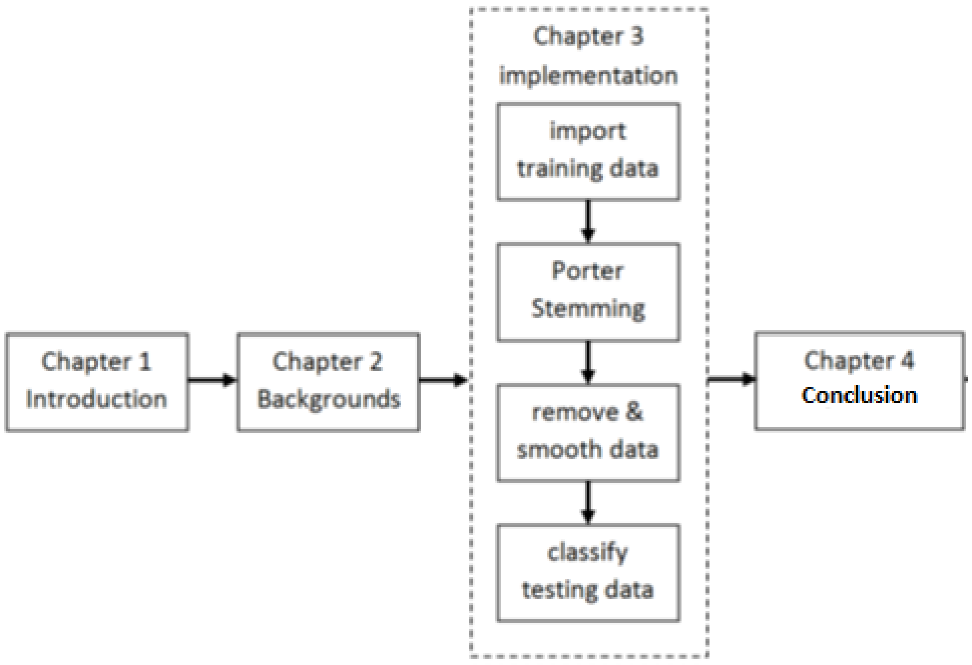
- Supervised machine learning
- Porter's stemming algorithm
- Naïve Bayes classification algorithm
- Data Training algorithm
- Rabin Karp string search algorithm

1.2 Report outline

This report is divided into five chapters.

Chapter 1 outlines the motivation of the spam filtering, and the overview of a standard probabilistic filter.
Chapter 2 covers the backgrounds of the project, including Bayesian network, Naive Bayesian classifier and the Porter Stemming algorithm.
Chapter 3 constructs a break-down view of the spam filter implemented in this project, covering four transition states of the system from training to actual deployment.
Chapter 4 concludes with a summary of contributions and programming issues.

The logical progression of the project is graphically depicted as follows:

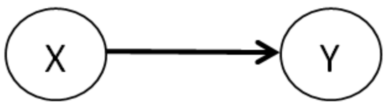


Chapter 2

Naive Bayesian Classification and Porter Stemming algorithm

2.1 Bayesian classification

A Bayesian network is a directed acyclic graph, which represents a set of nodes and their dependencies. A directed edge from node X to node Y as seen in figure below, means Y conditionally depends on X.



A node N is said to be conditionally independent of node M if N does not directly connect with M. Each node X is assigned with a probability table, which specifies the distribution over X, given the value of X's parent. Given a classification task, the Bayesian network can be applied to predict the decision outcome. For example, given the number of working hours and the stress level, the Bayesian network can represent the probabilistic relationship between number of working hours and the stress level. Then, if given a particular working hour number, the Bayesian classifier can work out the probability of the stress level.

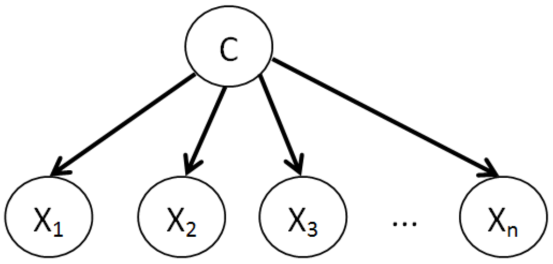
The standard formulae of Bayesian classifier is:

$$P(X|Y) = \frac{P(Y|X)\,P(X)}{P(Y)}$$

P(X|Y) stands for the probability of the event X, given the event Y.

2.2 Naïve Bayesian classification

Naive Bayesian classifier is simply the Bayesian classifier relaxed the dependency assumption. In particular, Naive Bayesian assumes that the presence or absence of any node in the Bayesian network does not affect any other nodes. For example, considering 'wet grass' and 'cloudy' as the two nodes of the network, although they both contribute to the probability of 'raining' event, the existence of 'wet grass' event does not affect the existence of 'cloudy' event and vice-versa.



The biggest advantage of Naive Bayes is the computational overhead reduction, in order to estimate a probability. The quantity P(X|Y) in the formulae in Bayesian classification is often impractical to calculate directly, without any independence assumptions. Since each node x_1; x_2; : : : x_n are conditional independent of each other, given a common class C, as depicted above, its probability can be calculated separately, and the combination of separate probability of each node can be combined to yield an overall probability of the big event. The general formulae for Naive Bayesian in terms of each separate node can be calculated as:

$$P(X|C) = P(x_1|C)\, P(x_2|C) : : : P(x_n|C)$$

2.3 Porter stemming algorithm

The Porter Stemming algorithm is widely used in the Information Retrieval area, due to the vast amount of data extracted from a document. The purpose of Porter Stemming is to group similar words, hence improving performance because the extracted data volume is reduced.

The algorithm takes any English word and returns its root form, by removing the word's suffix. For example, all three words 'teacher', 'teachers', 'teaching' convey the same base idea, and will be reduced to a shorter form 'teach' by the algorithm. Before examining details, it is worth noting that the algorithm is very careful not to remove any suffix when the word is already too short, to avoid losing the meaning of the word. Also the final result of the returned word is not necessarily meaningful in the linguistic form.

First, the definitions of 'vowel' and 'consonant' need to be clarified, as they are intensively used by the algorithm. A vowel in English is the letter A, E, I, O, or U. If the letter Y is not preceded by a vowel, it is considered as another vowel. Any of the letters which are not vowels are called consonant. For example, considering the word 'holy', the letter 'h' and 'l' are vowels, while 'o' and 'y' are consonant.
If a group of continuous vowels are denoted as V, and a group of continuous consonants are denoted as C, any English word will have one of the four following forms:

CVCV . . .C
CVCV . . .V
VCVC . . .C
VCVC . . .V

From the four above forms, a single united form can be defined as $[C](VC)^m[V]$, with [C] and [V] means C and V can be absent or present. For example:

with m = 0, an acceptable word is 'tree'.
with m = 1, an acceptable word is 'play'.
with m = 2, an acceptable word is 'teachers'.

To remove a suffix from an English word, the Porter Stemming algorithm applies the following rule:

$$\text{(condition) } S1 \rightarrow S2$$

This rule states that if the word satisfies the specified 'condition', and it also ends with the suffix S1, then S1 will be replaced with S2. Given a set of rules, the algorithm will prioritise the one with the longest matching S1. For example, given the following three rules:

(m = 2): ers → ing	[1]
(m = 2): s →	[2]
(m = 2): rs → s	[3]

The word 'teachers' satisfies all three conditions, however the rule [1] is applied first since it has the longest matching 'ers' suffix. The word 'teachers' will be replaced as 'teaching'.
The algorithm we have implemented has seven steps, in which a complicated long word is stripped bit by bit in each step to its root form. Each step contains a set of rules which strip all common suffixes such as plural forms, passive form, noun, etc.

Chapter 3

Filter Implementation

Having already discussed the Naive Bayesian classifier and the Porter Stemming algorithm, this chapter looks further to apply them in an actual spam filter. The chapter follows four transition states as the filtering system develops. First, the training data is imported to the system. The Porter Stemming algorithm is applied to every word in the training data to produce 'a token'. Then the new extracted tokens are passed through a small filter which removes any common words. Data is then stored in the training system and the final step simply queries the probability of each word required for classification.

3.1 Importing training data

This is the learning step, which teaches the filter which e-mail is spam and what is genuine. Two sets of training data are fed into the system. For each e-mail, the message content is broken down into smaller words. A word is a consecutive sequence of characters. Two words are separated by one or many spaces. The training data is encoded into a probability table. This table stores the probability of every word found in the training data, categorised into Spam and Non-spam classes. Thus, for some training data, a particular word will have a different probability for the Spam class and a different probability for the Non-spam class. To generate such table, the system first counts the frequency of each individual word in each class.

The formulae given below show how to calculate the probability of each word in the Spam and Non-spam class.

$$P\,(W_{new}) = \frac{P(W_{old})^{*}\,(\text{Counter}-1)+1}{\text{Count}}$$

$$P\,(W) = \frac{1}{\text{Counter}}$$

The variable- ‘Counter’ stores the number of spams and non-spams, respectively, during their training period. The formula written first calculates the probability of the word that already exists in the training set database, and the second calculates the probability of the word that doesn’t exist in the database.

3.2 Removing and smoothing data

There are many 'stop words' in English. These are the most frequent words such as 'the', 'a', 'of '. Their presence doesn’t contribute to the primary meaning of the sentence in particular, or the whole message's content in general. Not only do they increase the computational overhead, they also affect the final result caused by accumulating small errors. The opposite of those 'stop words' are the 'rare words'. These words appear so rarely in the training data and do not help the final result either. So, we have implemented Rabin Karp string search algorithm with hashing technique to solve this issue.

3.3 Porter stemming application

The Porter Stemming module receives a word and returns a new token in its original form, by stripping out the redundant suffixes, as discussed previously in chap- 2. By returning the words to their root form, the training data size can be greatly reduced, as well as increasing the probability of the word, hence resulting in a better classification. The algorithm is applied by the system when any new word is presented. The training data only stores the processed tokens returned by the algorithm.

3.4 Classifying a new e-mail

When a new email which needs to be classified, is presented to the filtering system, it is broken down into smaller words. The Porter Stemming algorithm is applied to these words. A question arises here: what if the original words are passed to the system without applying the Porter Stemming. The answer is the probability of the word will be extremely small, because the original form of the word is not recorded in the training data. In this case, these words do not contribute to the overall probability of the whole message.

The message processing in this step is similar to the one at the importing training data in step 1. However, to improve the system running time, the small filtering process to remove the most frequent words and rare words is ignored. This does not affect the results at all, because the system returns an extremely small probability for any frequent words appearing as they are not found in the training data.

By putting all processed words together, every e-mail can be presented as a vector ($a_1; a_2; : : : ; a_n$), with n is the total number of tokens in an e-mail. According to the Bayesian network formulae, given two class labels SPAM and NONSPAM

$$P(Spam|(a_1, a_2, \ldots, a_n)) = \frac{P((a_1, a_2, \ldots, a_n)|Spam)P(Spam)}{P((a_1, a_2, \ldots, a_n))}$$

$$P(Nonspam|(a_1, \ldots, a_n)) = \frac{P((a_1, \ldots, a_n)|Nonspam)P(Nonspam)}{P((a_1, a_2, \ldots, a_n))}$$

P (Spam | ($a_1; a_2; : : : ; a_n$)) states the probability this e-mail spam, given a vector ($a_1; a_2; : : : ; a_n$), similarly to P(Nonspam | ($a_1; a_2; : : : ; a_n$)). To decide if an e-mail is spam or genuine, the system compares the two terms. If P (Spam | ($a_1; a_2; : : : ; a_n$)) >P (Nonspam | ($a_1; a_2; : : : ; a_n$)), the e-mail is identified as spam and vice-versa. Besides, as P(Spam), P(Nonspam), and P (($a_1; a_2; : : : ; a_n$)) are constants, the problem becomes finding which one is larger between P (($a_1; a_2; : : : ; a_n$) | Spam) and P (($a_1; a_2; : : : ; a_n$)) | Nonspam).

Since each word in the e-mail is conditionally independent, given the class label SPAM or NONSPAM, the Naive Bayesian classifier says that:

$$P((a_1, a_2, \ldots, a_n)|Spam) = P(a_1|Spam)P(a_2|Spam)\ldots P(a_n|Spam)$$

$$P((a_1, a_2, \ldots, a_n)|Nonspam) = P(a_1|Nonspam)\ldots P(a_n|Nonspam)$$

Each small term P(a_i|Spam) and P(a_i|Nonspam) can be obtained directly from the probability dataset that we have created.

Chapter 4

Conclusion

In this project, we implement a spam filtering system using Naive Bayesian classification with the Porter Stemming algorithm. Besides, we perform an in-depth analysis of the filter performance in terms of safety, efficiency and running time. The actual effect of Porter Stemming on the system performance is also investigated.

In general, all above goals are achieved. In this chapter, we present other practical programming issues facing through-out this project, as well as an outline of the scope for future work.

4.1 Programming issues

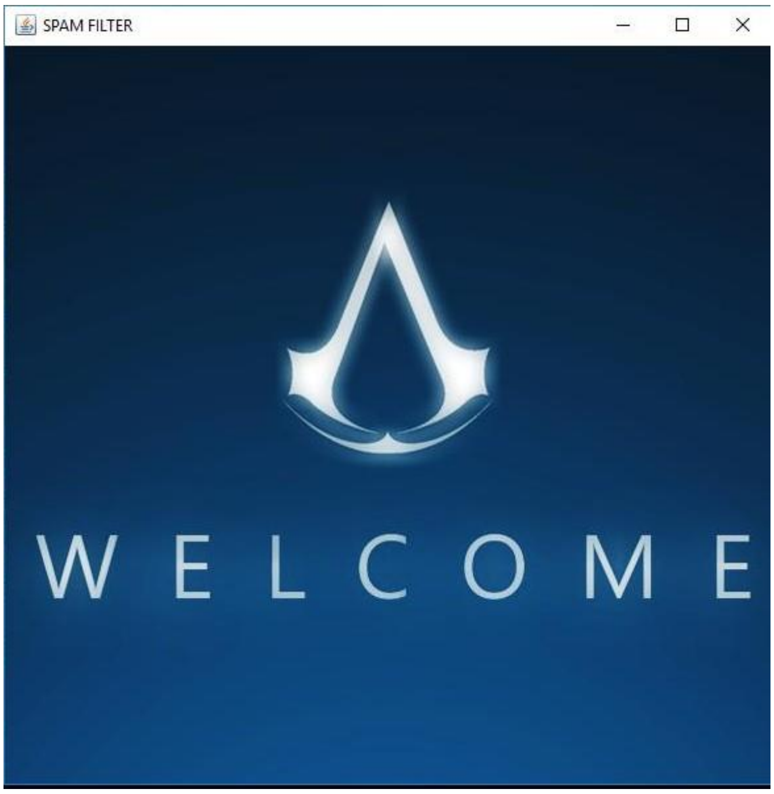
When the training data is not sufficient, a word's probability might not be found in the table. Sometimes the probability of a word differs in its 15th or more place of decimal representation and since the limit of decimal support is only 15 decimal places, so to prevent the whole equation turning into zero, we use Big Decimal class of Java with whose use, we can set the limit of decimal digits and store as many decimals as the limit of string i.e. as the length of 10^6.

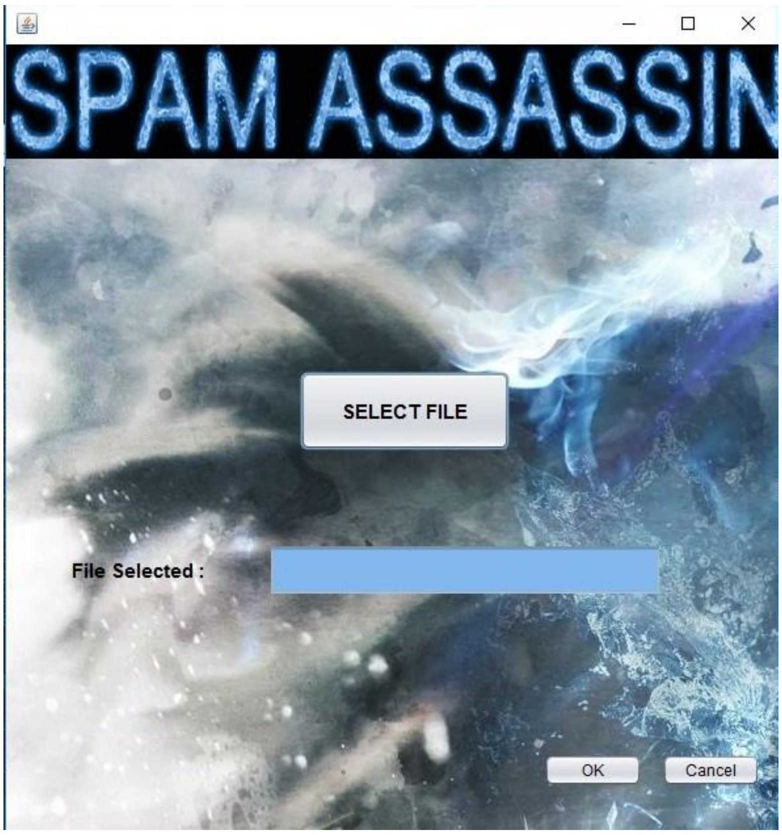
References

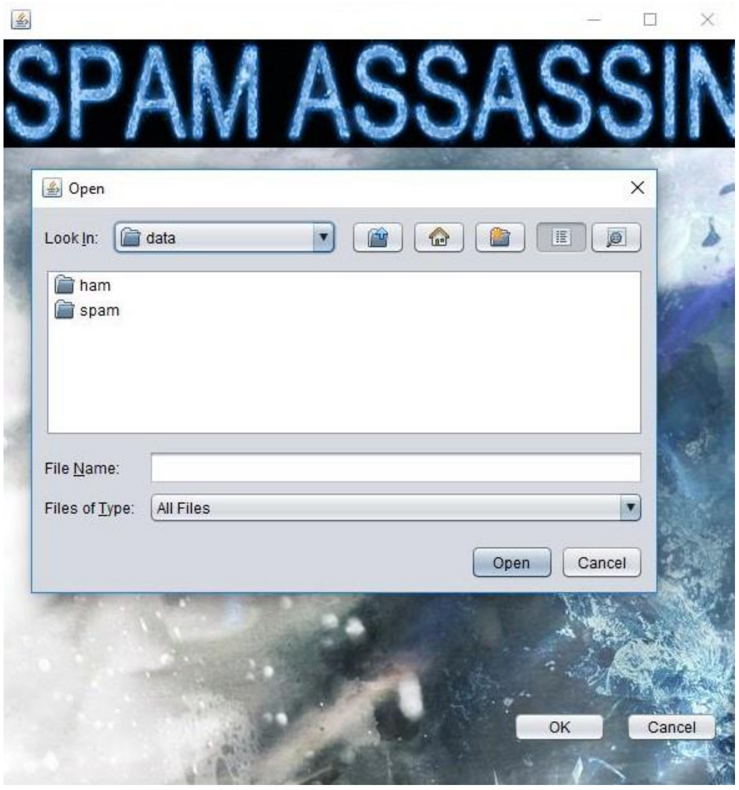
- Spam Detection-A Bayesian approach to filtering spam
  - by Kunal Mehrotra & Shailendra Watave.

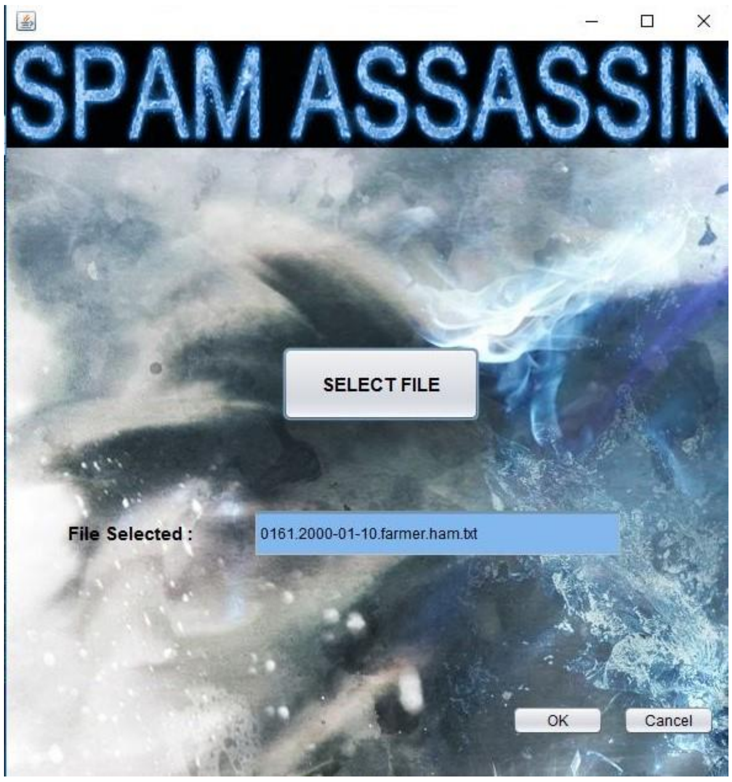
- Spam Email Filtering
  - by Shahar Yifrah & Guy Lev

Snapshots

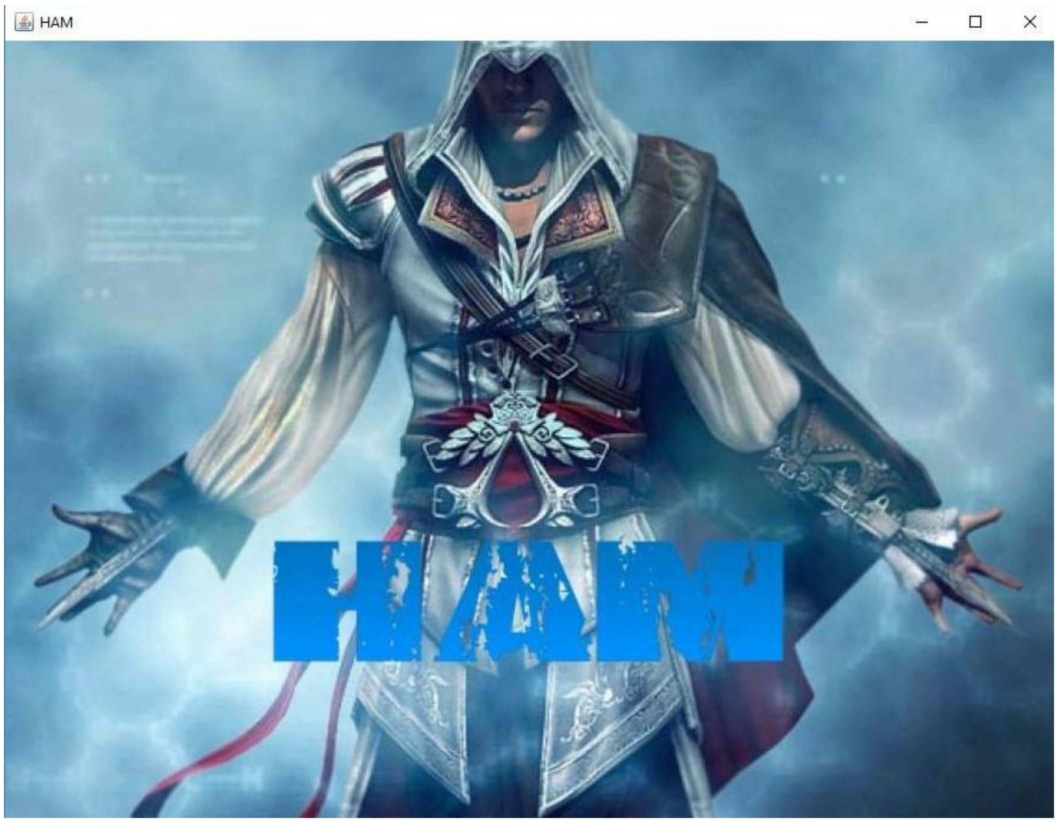

SPAM FILTER
WELCOME


SPAM ASSASSIN
SELECT FILE
File Selected :
OK
Cancel





File Selected : 0161.2000-01-10.farmer.ham.txt



----------xxxxxx----------