Autonomous Software Installation using a Sequence of Predictions from Bayesian Networks

Behraj Khan, Umar Manzoor, Tahir Syed
National University of Computer and Emerging Sciences, Karachi, Pakistan

***Abstract*—The idea of automated installation/un-installation is a direct consequence of the tedious and time consuming manual efforts put into installing or uninstalling multiple software over hundreds of machines. In this work we propose what is to the best of our knowledge the first learnable method of autonomous software installation/un-installation. The method leverages text classification using as data textual guidelines given for users on the installation window. This is used to arrive at the $Next/Pause/Abort$ decisions for each installation window using multiple classifier schemes. We report the best results using a full Bayesian Network with accuracy level of 94%, while Naïve Bayes and rule-based inference accuracy was 42% and 88%. We attribute this to the sequential nature of the Bayesian network that corresponds to the sequential nature of natural language data.**

***Keywords*—Multiagent System; Machine Learning; Software installation/un-installation**

I. Introduction

With the evolution in distributed environments both network size and complexity have increased substantially. The task of maintaining and improving a network generally requires multiple software installation and un-installation processes. This is greatly dependent on manual effort and therefore scales poorly. Consider the whole process of installation/un-installation which starts with initializing and running the setup wizard, at every screen/step reading the message/text displayed, analyze that text and then choose the appropriate action. This process is repeated till the installation/un-installation finishes successfully. Some software provides the option of silent or unattended installation in the form of set of switches [4]. *Silent installation* is the one which does not require interaction with the user at every screen/step to proceed further. Rather, it will be initialized by the user in the beginning and rest of the action sequence will be performed by the application itself.

However, silent installation proves to be a non-desirable choice for a distributed environment. This kind of silent installers are software- and vendor-specific. The degree of required human computer interaction for accessing every machine and initializing the setup is also very high. In addition to that this silent installation feature is provided for installing software only; to uninstall software no such facility is supported. Not all the software provides the silent installation option. The focus had majorly been upon automated installation on a standalone system within a non-distributed environment.

Automating installation/un-installation in a distributed environment where there might be several sub networks interlinked to each other is a complex and difficult goal to achieve as it increases the size and complexity of the autonomous framework. It may also involve the complexity of finding path over the relevant sub network to reach the destination node, than transferring files to that node. The process of safe and reliable transfer of files precedes the verification and initialization steps which are followed by running the setup and finally it concludes with the update in the system directories and sending the acknowledgment.

Some frameworks for silent/automated aid for software installation / un-installation has been proposed by U. Manzoor and S. Nefti (NDMAS [1], ABSAMN [2] and SUIPM [3]). These models were based upon rule based analysis of the text which is not very efficient and effective in unknown environments.

The installation/un-installation procedure is traditionally a resource dependent task and requires much of manual aid. To lessen this manual dependency and effort this task could be assigned to multiple intelligent agents with efficient learning and text classification capabilities. We automate the process of installation through setup wizard into silent installation. In this paper we propose a framework for installing a software without human intervention, i.e. by automating the process of the so-called silent installation/un-installation. *This is done by interpreting the text that appears on installation screens and thereby predicting the action to be taken next.* In our proposed framework the installer agent (the artificially-intelligent agent which will install software autonomously) activates once the particular installation file is run. The agent activated it extract all the information on application window and on UI controls (e.g. text on buttons like "‘Next"', "‘Back"', "‘Cancel"'). That helps the installer agent classify the text on the windows using classifiers such as a Bayesian network and thereby decide to continue installing the particular software or to quit installation. To summarize, our method works in the following manner:

- Access installation package and network nodes' resources,
- Read the text on an installer window,
- Classify the text using a number of classifiers and decide between classes 'Next' versus 'Cancel'.

Therefore, this work represents the use of text classification as a means to address a problem in distributed assisted software installation/un-installation.

The rest of the paper is organized as following: Section 2 presents the related work in the field, Section 3 presents the system architecture that would serve as a guide for the following sections that describe the algorithms that plug into components of this architecture, and in section 4 classification models are presented. Section 5 and 6 will focus on experimentation and analysis followed by conclusion.

II. Related Work

The idea of intelligent installation/un-installation agents has received attention from a niche group within the machine learning community. Manzoor & Nefti implement multi-agent aided network monitoring and installation application like "SUIPM", in which they proposed silent unattended installation package manager which generates silent unatteneded installation packages before installation. SUIPM supports all kind of software installation over the heterogeneous setting on different nodes. SUIPM does not require any client software for installing a particular software. However the proposed method has no intelligence, requiring initial training for the operator for installing software. SUIPM generates its own packages for installation which may be twice in size of the original setup[1].

Manzoor & Nefti propose a method in Cognitive Agent for Automated Software Installtion "CAASI"[2] that rectifies the shortcomings on [1]. The proposed method is able to install a particular software intelligently and the setup size of particular installing software remain as original setup, but the proposed method still required training before installing a software.

Manzoor & Nefti propose "ABSAMN"[3], which is an agent-based architecture for activity monitoring over the network. The proposed method watch activity monitoring like user activity, node level activity, and internet monitoring autonomously.

Manzoor & Nefti proposed a framework Smart network installer and tester for installing software autonomously "SNIT". SNIT is also agent based and motivated by An agent based system for activity monitoring on Network "ABSAMN". The proposed method supports unattended installation over the network intelligently. The proposed method install a software intelligently without any kind of training. SNIT do not require any specific kind of setup before installation.

Herrick & Tyndall [22] proposed a method Sustainable Automated Software Deployment Practices for automating software installation SASDP. but the proposed method have no intelligence and user have to watch the installation process till completion. SASDP requires a particular MSI software for running the application, and also give manual installation facility to user.

In the above applications, rule-based text classifiers were implemented for the learning of agents. The short comings of rule based classifiers are observed to be large and slow knowledge-base along with the inefficient performance in unknown environments.

Installation agents may learn through text classification schemes. Text classification is a challenging domain because of a vast number of attributes in the form of words. Many of the efficient text classifier models are designed for the domains with relatively short vocabulary set. But the more practical scenarios usually consist of complex and large vocabulary set (more than thousand words). Many text classification schemes have been proposed and implemented for large vocabulary sets (a detailed comparison has been presented in the table 1). For our research we will classify the text using the Naïve Bayes and Bayesian belief networks.

In their work, Domingos and Pazzanihas [4], [5] concluded that Naïve Bayes based text classification promises very optimistic results with the constraints of zero significance level of the probability calculated by the Naïve Bayes . Some literature shows the implementation of Naïve Bayes classifiers using a binary feature vector. The implementation does not capture the total occurrences of a word in the text rather it captures the probability of all attribute values (both present and absent). A binary feature vectors represents absence or presence of every word.

Therefore, the text is modeled as an event with related binary attributes. This category of implementation is called multivariate Bernoulli Naïve Bayes. This model is closest to the traditional Naïve Bayes. The shortcomings of this implementation is its inability to exploit word frequencies in the text and its suitability for tasks with fixed number of attributes (small documents).This approach has been applied for various text classifiers.

The other Naïve Bayes text classifier implemented is the multinomial model or the standard Naïve Bayes text classifier [6], [7], [8], [9].This model focuses on the number of occurrences of a word, showing the words as events. The order of appearance becomes insignificant and the probability of a particular word becomes significant.This approach has also been applied to multiple domains of text classification like speech recognition and spam filtering.

The standard Naïve Bayes text classifier does not show better performance in comparison with other statistical learning methods like support vector machines [11], nearest-neighbor classifiers [8], and boosting [10]. However, the shortcomings of this model are its rough parameter estimation.Latest researches have focused on exploiting the efficient and simple implementation scheme in multiple practical domains of text classification like web mining and news article classification.

A Bayesian belief model focuses on relationship among the attributes. It could also be applied with different statistical methods to gain enhanced performance and avoidance of data over fitness. It shows good results even if some data entries are missing as it models the dependencies amongst all attributes. This model could be applied to gain better understanding of the problem domain. It is widely used in different application areas of classification. These networks can be learned automatically on the basis of statistical analysis. Naïve Bayes is a special case of these networks. Many text classifiers based of Bayesian belief networks has been introduced [12][13] for multiple domains like disease and cancer diagnosis.

III. Proposed System

The proposed framework will work in a layered fashion/architecture. The top most layer in the hierarchy is the supervisor layer, that controls the whole framework. The supervisor layer controls the controller layer which controls the verification layer. The lowest layer in the framework is the installation/un-installation layer.

A typical distributed environment could be seen as a network of multiple networks. A supervisor agent is moni-

TABLE I. COMPARISON OF TEXT CLASSIFIERS

Text Classification Model	Application Domains	Advantages and disadvantage
Decision Trees [22]	Hierarchical distribution Skewed class patterns Predicate based, divide and conquer strategy	Simple knowledge representation Fast learning and qualitative analysis Inefficient when there is noise in the training data [5]
Rule based classifiers [24]	Based on simple rules to represent text categories Classification rules are defined manually Decision support systems	Easy to understand and modify Easy incremental update by other machine learning models Use of more than one feature values simultaneously Inefficient with the exponential growth of feature space Large number of training rules [25] Conflicting rules and low coverage Change in rules with the change in an environment
SVM Classifiers [26]	Supervised classification algorithms Partitioning of data space into different classes	No transformed space is required Consistent solutions Focuses on linear separators and robust over fitting applicable to binary classifications only frequent generation of zero values Time consumin [27]
Neural Network Classifiers [28]	Multi-Output Perception Learning algorithm (MOPL) Back-Propagation Neural Network (BPNN).	Quantitative analysis Complex knowledge representation, Slow learning and converges on local minima For long trainings it starts over fitting Multiple training runs are required for assessing the applied model [Yao and Zhi-Min et al., 2011],[Manning et al., 2009]
Naïve Byes classifiers	Attributes are considered as independent Web mining, news group classification	Simple, fast and efficient Cheap implementation cost
Bernoulli Naïve Bayes	Speech recognition, Web mining, Spam filter etc	Restrictive conditional independence, poor performance for strongly correlated data [5],[5]
Multinomial Naïve Bayes		lack of uniformity in training data [24] some attributes might remain less trained in comparison with other
Bayesian Belief networks	Attributes are considered correlated to each other Missing attributes could be computed	Fast and efficient Delivers better understanding of the domain knowledge

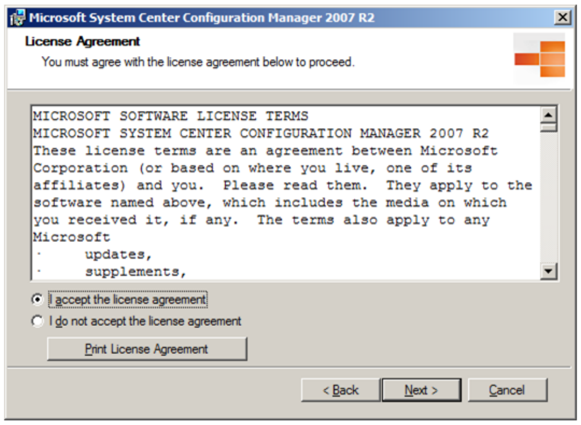
toring the whole framework at the master server level and multiple controller agents are assisting and coordinating with it .Whereas every controller agent is monitoring a separate sub network. Multiple file transfer agents will serve the purpose of IS-UIS file transferring over the path on the network from source to destination node under the supervision of their respective controller agent. The main and primary objective of the application is to design classifiers to support the installation/Un-installation tasks and to update the knowledge base of the system. The application/program to be installed/un-installed will be pre-processed by the system agents. This task involves gathering the information on every screen as given in figure 1 below:

The information on the screen is in the given format:

- Screen label
- Text/data
- Buttons/text boxes/ check boxes/ option groups along with their text ...

The data (above mentioned) will then be transferred to the classifiers and then data will be processed to make it ready for the classification task as given in figure 2 below:

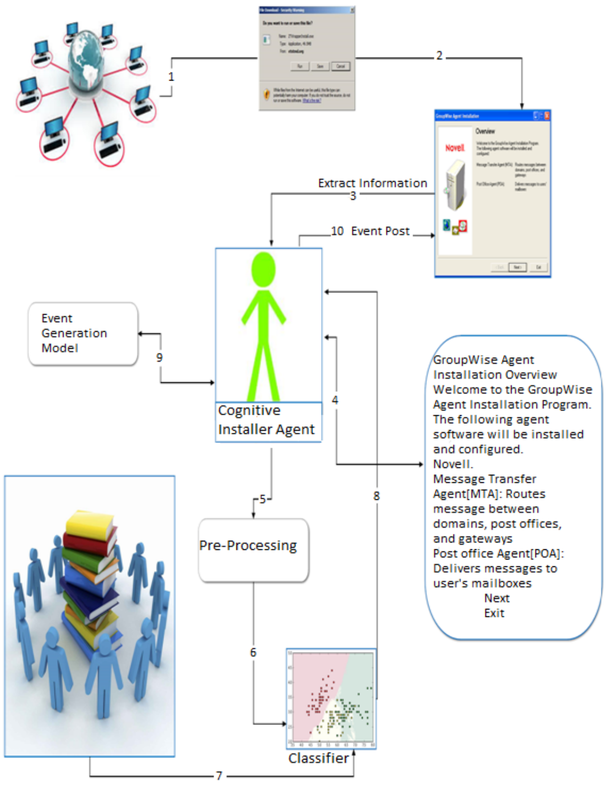
The information on the software installation window will be extracted first from that window which contain standard amount of text, after extracting data will be pre-processed means that the words like (a, an, the, are, is, of, to, will) be removed from data and then will be given to classifier. The classifier would have a knowledge base on basis of which


Microsoft System Center Configuration Manager 2007 R2
License Agreement
You must agree with the license agreement below to proceed.

```
MICROSOFT SOFTWARE LICENSE TERMS
MICROSOFT SYSTEM CENTER CONFIGURATION MANAGER 2007 R2
These license terms are an agreement between Microsoft
Corporation (or based on where you live, one of its
affiliates) and you.  Please read them.  They apply to the
software named above, which includes the media on which
you received it, if any.  The terms also apply to any
Microsoft
·       updates,
·       supplements,
```

[✓ I accept the license agreement]
I do not accept the license agreement

Fig. 1.   Gathering Information



GroupWise Agent Installation Overview
Welcome to the GroupWise Agent Installation Program.
The following agent software will be installed and configured.
Novell.
Message Transfer Agent[MTA]: Routes message between domains, post offices, and gateways
Post office Agent[POA]: Delivers messages to user's mailboxes
Next
Exit

Fig. 2.   System srchitecture

it would classify the text and would give the result to event generation model, event generation model would generate an event and then event will be posted.

A feedback module has been added to the application to enhance the accuracy of test cases and to achieve more accurate and promising results for the test data. The feedback module starts working after the completion of classification and it asks the user to help identifying the unidentified cases by giving the class labels as input as given figure 3 below:

The knowledge base is updated according to the user input and whenever new data is tested, the existing knowledge base is also referenced. The processing data will be divided into classes and attributes (tokenized) in order to find the probabilities of relevant attributes according to their expected/classified class labels. A knowledge-base will be maintained to break down the attributes from the data. That knowledge base of data attributes will be used by the three classification algorithms to compute the probabilities and relationships. The classification task comprises of two steps:

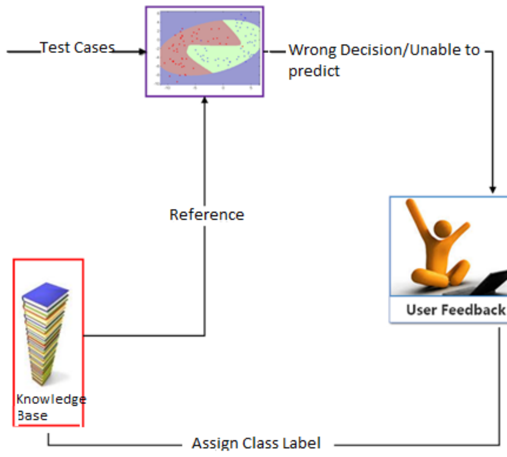


Fig. 3.   Feedback Module

- Classification of class label,
- Identification/selection of appropriate action.

Once a screen/attribute has been classified correctly, the objective is to identify the appropriate action/button selection for that particular screen/attribute. Knowledge-base will be maintained by the application to keep track of the expected desired options to be selected/opted by the user against each class to help identifying the action/button to be finalized.

A. *Agent Infrastructure*

In an autonomous environment the role and behavior of an intelligent mobile agent has always been an ideal choice to represent flexibility and reliability towards achieving the design objective in a distributed environment. The aim of this research is to present an agent based autonomous framework for software installation/un-installation in a distributed environment. The agent based autonomous software installation/un-installation framework will conduct the automated installation/un-installation task over a network. The intelligent software installation/ un-installation agents will be trained using Naïve Bayes and Bayesian Belief classifiers. The system consists of five agents:

B. *Server Agent*

Server agent is the main agent of the system it initializes the whole system. SA loads the network configurations which contains the IP address and name of the sub server and it also contains the range of the sub server network on which particular software to be installed.

C. *Sub-Server Agent*

Server Agent initiate sub server agent, then sub sever move to each sub server and perform the following steps. Sub Server

Agent load the configuration file and also load the Knowledge Base. Sub Server Agent is also responsible for creating and initializing File Transfer Agent and Installer Agent.

D. File Transfer Agent

File Transfer Agent is responsible to transfer the software to each node over the network. It distributes the file from a file server to a large number of machines over the network. FTA (File Transfer Agent) transfers the software into chunks and then merges it on each node.

E. Installer Agent

As the FTA completes their task then Installer Agent is initialized. IA contains the list of nodes on which particular software to be installed. Then it moves to first node in the list, loads the Knowledge Base, software profile and also the information passed by the Sub Server Agent. Before starting the installation, Installer Agent will fist check the constraints like space available in the target drive or not.

F. Verifier Agent

After initializing verifier Agent (VA), VA moves to each node and verify that the product installed on each node. After verifying the information about product VA will run the application in back ground and then moves to other node. The method proposed by Umar et al. [19] installed software over the heterogeneous network autonomously. SNIT installs the software(s) over the network autonomously, but the proposed method is rule based and failed in some cases. To overcome this problem we proposed a method in which we extend the SNIT and proposed autonomous installation using Bayesian Belief Network. In this thesis I only modify the Installer Agent of SNIT and named them Cognitive Installer Agent (CIA). CIA installs software over the network autonomously using Bayesian Belief Network. CIA is more intelligent than installer agent, and it can handle all type of software installation and un-installation without any user interaction.

IV. Text Classification Models for the Installer Agent

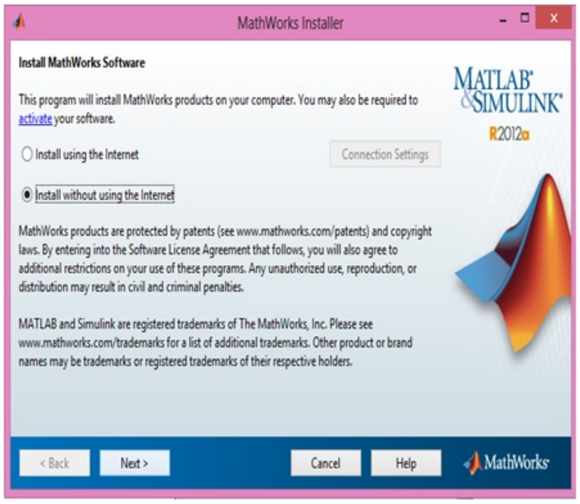
Different installation packages may have differently-worded text showing on installation wizard screens, but at the same time there are many similarities among the text displayed, beginning with the standard text on UI controls like buttons and keywords that on their own may be sufficient for classification under the iid ssumption that for instance Naïve Bayes takes. So for the task at hand we work with a number of successful classifiers from the text retrieval community like Naïve Bayes, Bayesian Belief Network, and rule-based classifier and compare their performance.

A. Naïve Bayesian classifier

Naïve Bayesian classifier are used commonly for text classification because of its success in this field and its ability to work with limited amounts of data. In Naïve Bayesian classifier we attempt to make a probabilistic classifier which is based on molding the fundamental word features in different classes. The main idea behind is to classify the text on the base of posterior and prior probability. In nave Bayesian classification we trained our classifier in such a way that the target class is known, and then we test the classifier for unknown target class. The classifier has to learn on the input data along with output data.

With the Naïve Bayesian classifier, we first calculate the probability of the target class in the document in the training set and then the probability of each attribute with respect to the target class, after that when the new data is provided to the classifier as test data then it assigns the data target class which probability is higher [3]. The Naïve Bayesian classifier gives the precise result when it is provided a large number of data set. Still it is often difficult that we have a large number of data sets. Beside these when the datasets are large then it required more space and have more time complexity for running. So the case in which we have small data set as is our problem then it give us quick result.

Classification of text is a becomes more challenging with the amount of data present. But classification of large amount of text is much easier than small amount of text document. Our problem is very challenging task because in this particular problem amount of text is very small as text on installation software text window which is very small in amount and consists of standard keywords as given in figure 4 below:


MathWorks Installer

Install MathWorks Software

This program will install MathWorks products on your computer. You may also be required to activate your software.

( ) Install using the Internet   Connection Settings

[✓ Install without using the Internet]

MathWorks products are protected by patents (see www.mathworks.com/patents) and copyright laws. By entering into the Software License Agreement that follows, you will also agree to additional restrictions on your use of these programs. Any unauthorized use, reproduction, or distribution may result in civil and criminal penalties.

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. Please see www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Fig. 4. Installation Window

The text extracted from this window is:

```
Caption
MathWorks Installer
Labels
Install MathWorks Software
This program will install MathWork products on your computer.
You may also be Required to activate your software.
Radio Buttons
Install using Internet
Install without using Internet
Button
Connection Settings
Labels
MathWork Products are protected by patents
(see www.mathwork.com/patents) and copyright laws.
By entering into the Software License Agreement that follows,
you will also agree to additional restrictions on your use
of these programs. Any unauthorized use, reproduction, or
```

```
distribution may result in the civil and criminal penalties.
MATLAB and Simulink are registered trademarks of The
MahthWorks, Inc. Please see www.mathworks.com/trademarks for
a list of additional trademarks. Other product or brand names
may be trademarks or registered trademarks of their respective
holders.
Buttons
Next
Cancel
Help
:
```

As we discussed in previous section that the text on the installation wizard window is small and consists of standard keywords.Now the problem is how to classify this text. Among different types of classifiers which we studied earlier the most suitable classifier for classifying the smaller and standard text is the Bayesian Belief Network which gives us better result than others.

After initialization of co-installer agent, filtered text (standard keyword) of particular window will be passed to classifier. The classifier represents the received information as given in figure 5:

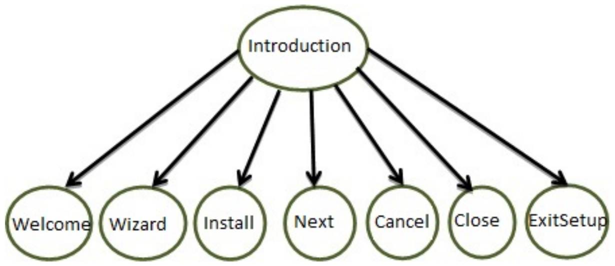


Fig. 5. Naïve Bayes

Above is the model for the very first window of installation process which is Introduction. After classification the result is given to event generation model and then other screen of setup wizard provided to the classifier autonomously and a description similar to the figure is generated for each of them.

B. Bayesian Belief

Bayesian Network is a probabilistic graphical model that represents a problem into set of random variable. In Bayesian Network model a problem is represented in the form of directed acyclic graphical model in which every node have probability. In directed acyclic graph, nodes represent variables and edges represent the probability among nodes. Each of variable have some variables (nodes) on which it depends that is called parent of them. Each of variable have their parents. Besides these Belief Network also has a probability table which shows the probability of each child with their parents. The conditional probability (CPT) of X and its parents is represented by a clique of size (k+1) in the graph and have $d_k(d-1)$ parameters. Learning process in Belief Network consists of two parts one is learning the network graph structure and other is learning the probability [13]. Bayesian Networks have three types of connections serial, diverging and converging.

In Bayesian learning the network is learned on trained data which contain output classes and then data without output classes are given to the network for testing[6].

The Bayesian network based classification is gaining popularity in almost every field of evolutionary sciences. It has been found very useful especially when the data is missing and incomplete. The Bayesian based classification has different forms. The focus of the research is performance measuring of Naïve Bayes, Bayesian Belief and Tree Augmented based classifiers and their comparison on the installation/un-installation data. Following is a brief description of the working of each of the three models along with the design details.

In our proposed design as the co-installer starts initialization it takes the extracted information from the window screen and provide it to the Bayesian belief classifier. Bayesian belief classifier represents the received information in network like structure and calculates the probability for the provided screen and passes it to event generation model. Following figure 6 is a brief model for the very first screen of setup wizard.

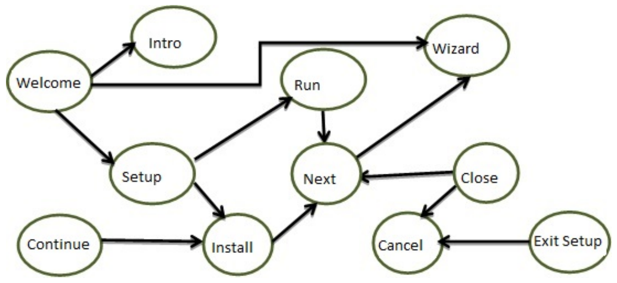


Fig. 6. Bayesian Belief

After classifying first screen, remaining windows of installation process are provided to the classifier in same fashion.

C. Tree augmented Bayesian

As co-installer initialized it takes preprocessed text of the very first window screen of the installation setup wizard window and provide it to augmented tree for classification. Augmented tree takes the received information from co-installer agent and represent it in tree like structure as shown in figure 7 below for classification.

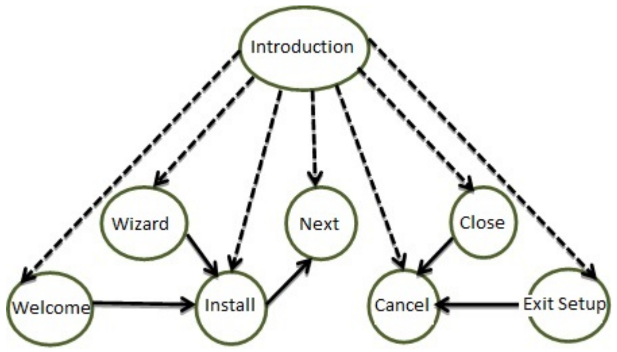


Fig. 7. Tree-augmented Bayesian Belief Network

Above is a brief model for very first model, same is generated for every window screen after receiving from co-installer agent.

V. Experiments and Discussion

The classification models were trained for exemplars taken from 25 different installation software programs and then tested for unknown test data set collected from 15 other software; a ratio of 62% training data set and 38% testing data set. Over all the results of the Bayesian Belief are found very promising. Following is a brief description that illustrates the performance of both the classification techniques with respect to different attributes.

Overall, the Naïve Bayes based classifier shows 42% accuracy rate with 84 exemplars classified correctly, rule-based showed 88% accuracy with 174 exemplars classified correctly, while the Bayesian Belief model classifier outperforms the others with an accuracy level of about 94% by classifying 187 cases correctly. The failure rate in terms of incorrect classification is 58%, 12% and 6% for Naïve Bayesian, rule-based inference and Bayesian Belief based models respectively.

In terms of training and testing data set; the Naïve Bayes based classifiers show 60% and 38% accuracy, rule-based inference shows 87% and 13% accuracy for training and testing data set respectively. Whereas the Bayesian Belief classifier performed very well by correctly classifying 96% and 92% of training and testing dataset respectively. The graph given below represents the overall performance presented by both the classification models.

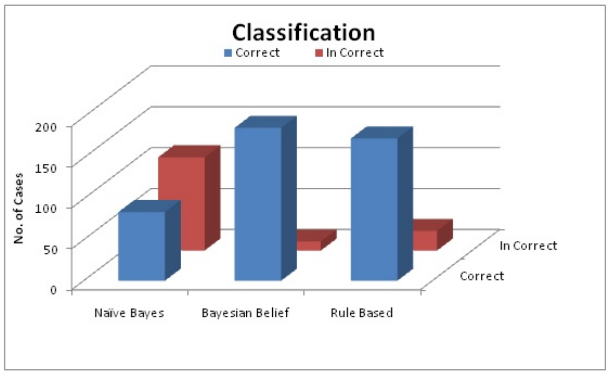


Fig. 8. Performance comparison of Naïve Bayes, Rule-based and Bayesian Belief Network

The dataset is categorized both by classifiers and also by rule-based inference. There were a total of 21 classes; each containing their relevant set of exemplars. We also measure the performance of the rule-based, Naïve Bayesian and Bayesian Belief classifiers for each class label. As discussed above the Bayesian Belief model was found to be more accurate and promising for each class (both training and testing data). The class design was based upon the output action selection attribute; Run, Next, Finish etc. As Bayesian Belief model focuses more upon attribute dependencies it performed well. On the other hand the Naïve Bayesian model assumes no dependencies amongst the data and depicted poor performance.

Given in Fig. 9 below is the graph showing the performance comparison of both the classifiers for 13 classes out of 21 as they demonstrate the good and true representation of all the classes (the similar trend was observed in the remaining classes as well). The total number of exemplars per class varied as the data was taken from installation software screens.

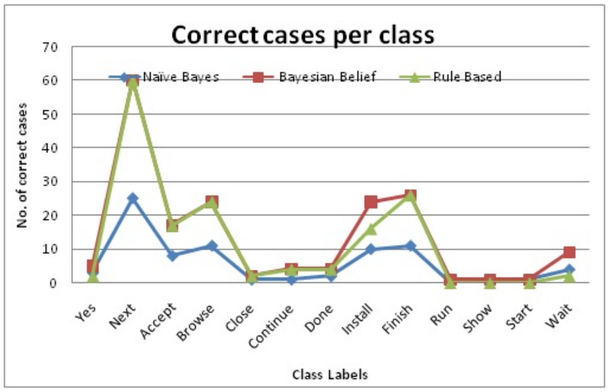


Fig. 9. Analysis of correct cases per class

As discussed above the screens/cases of a total of 40 software were used as the dataset for our research. Each individual screen while running software was considered as an individual and distinct exemplar. We collected a total of 198 exemplars for our experimentation. For a probabilistic classification paradigm this figure/number is considered to be sufficient. To measure the performance of any classification model the accuracy of results are the primary concern followed by the time constraints as the secondary point of focus. The time complexity of a Naïve Bayesian model is simpler and less than that of Bayesian Belief model and Rule Base in terms of time taken as well as the number of steps involved in the classification process.

Another comparison to judge the time based performance of both the classifiers and Rule Base system is to compare the time taken to classify/ predict single independent installation software. A total of 8 distinct software were selected and classified using both the models and also on Rule Base to measure the time taken to classify all the screens/dataset for each. Once again due to its simplicity of relationship and absence of inter dependency amongst the attributes the Bayesian Belief model showed better results.

Fig. 10 shows the graph representing a comparison of time efficiency of both the classification models and Rule Base for 8 different installation software.

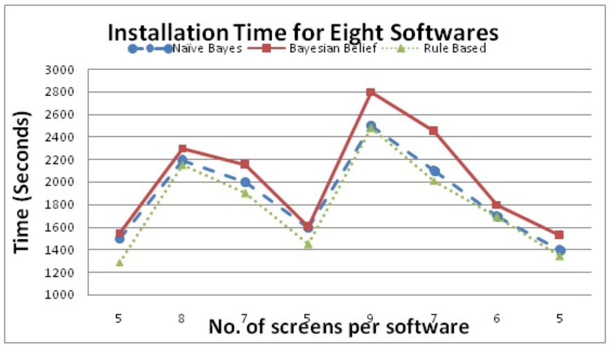


Fig. 10. Analysis of time

The whole research is focused upon the concept of automated installation; the support to this prime task is essential to be discussed. Hence their performance while running software was also judged along with measuring the outcome of both classifiers, Randomly 5 different installation software were selected and the screens/data set from all of them were classified using both the models. The results of the Bayesian Belief model were amazing; in some software it showed even 100% performance by selecting the correct output class. Whereas the output of Naïve Bayes model and rule-based inference was disappointing as in some software it could not show more than 40% correct classfication. During installation Bayesian Belief Model is found to be resource-hungry in terms of time as compare to Naïve Bayesian and rule-based inference,which may affect system scalability once deployed, but our main focus is on accuracy of the number of softwares that are correctly installed. So Bayesian Belief Model is the method of choice available. We surmise that the major reason for its success is the fact that it does not ignore temporal relationships (i.e. one word following a certain number of others) between variables.

VI. Conclusion

In this chapter we compare the result of different classifiers like Naïve Bayesian, Bayesian Belief Network, augmented tree and rule base classifier. We also generalize in this chapter that which classifier performance is best than others. The proposed frame work is unique to the best of our knowledge as it will guide the installation/un-installation setup on the basis of information retrieved from the knowledge base; designed and modeled upon the Naïve Bayes and Bayesian belief classifiers. The autonomous framework is capable of handling unexpected situations and responds to the changes in the environment during the execution. In addition to this, it is capable of learning by adding new facts to the knowledge base. We also validate an implementable system architecture or framework, based upon a multi-agent environment interact and cooperate with each other in order to meet their design goal [21, 22].

References

[1] Charu C. Aggarwal and ChengXiang Zhai. 2011. A Survey of Text Classification Algorithms, Pattern Recognition, volume 37, Issue 3,(28 May 2011): pp 169-180, DOI: 10.1007/978-1-4614-3223-4-6.

[2] George Tsatsaronis and Vicky Panagiotopoulou. 2011. A Generalized Vector Space Model for Text Retrieval Based on Semantic Relatedness, (2009):Pages 70-78

[3] Gerhard Weiss. 1999. Multiagent Systems: A Modern Approach to Distributed Modern Approach to Artificial Intelligence (USA: MIT Press, 1999) ISBN 0-262-23203-0.

[4] H. Van Dyke Parunak , Paul Nielsen , Sven Brueckner and Rafael Alonso. 2007. Hybrid Multi-Agent Systems: Integrating Swarming and BDI Agents, Volume 4335, (2007), pp1-14.

[5] Han-joon Kim1 and Jae-young Chang. 2003. Improving Naïve Bayes Text Classier with Mo died EM Algorithm, Volume 2871, (2003): pp 326-333, DOI: 10.1007/978-3-540-39592-845.

[6] Ian H. Witten, Eibe Frank and Mark A Hall. 2011. Data Mining: Practical Machine Learning Tools and Techniques.(USA: Elsevier, 2011).

[7] James Allen, Nate Blaylock and George Ferguson. 2002. A Problem Solving Model for Collaborative Agents.,(July 15-19, 2002), DOI:10.1145/544862.544923.

[8] James Ingham. 1997., What is an Agent?, Technical Report 6/99 (1997),.

[9] Kotz, David, Mattern, and Friedsmann. 2000. Mobile agent applications, Volume 1882 (September 13-15, 2000) ISBN 978-3-540-45347-5.

[10] Keith S. Decker and Katia Sycara. 1997. Intelligent Adaptive Information Agents, Volume9, Issue 3, (1997/11/12): pp239-260. Klaus Dorer, Applications of Multi-Agent Systems in Logistic:Lecture 10. Hochschule Offenburg University of Applied Sciences.

[11] Manzoor and Nefti. 2010.,QUIET: A Methodology for Autonomous Software Deployment using Mobile Agents. Volume 33, Issue 6,(November 2010): Pages 696706, DOI: 10.1016/j.jnca.2010.03.015.

[12] Manzoor and Nefti. 2011 " Autonomous agents: Smart network installer and tester (SNIT)." Volume 38, Issue 1,(January 2011): Pages 884893, DOI: 10.1016/j.eswa.2010.07.066.

[13] Manzoor and Nefti. 2009. An agent based system for activity monitoring on network, Volume 36, Issue 8 (October 2009): Pages 10987-10994, doi:10.1016/j.eswa.2009.02.060

[14] Peter Stone and Manuela Veloso. 2000. Multiagent Systems: A Survey from a Machine Learning Perspective, volume 8, Issue 3, (June 2000): pp 345-383.

[15] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schtze: An Introduction to Information Retrieval, page 181. Cambridge University Press, 2009

[16] Stuart J. Russell, Peter Norvig. 1995. Artificial Intelligence: A Modern Approach. (New Jersey: Prentice Hall, 1995), 07632.

[17] Sang-Bum Kim, Hee-CheolSeo and Hae-Chang Rim,Poisson Naive Bayes for Text Classication with Feature Weighting Volume 11,(2003): Pages33-40 doi:10.3115/1118935.1118940.

[18] Songbo Tan , Yuefen Wang and Gaowei Wu. 2011. Adapting centroid classier for document categorization, Volume 38, Issue 8, (August 2011): Pages 10264-10273, DOI: 10.1016/j.eswa.2011.02.114.

[19] Songbo Tan. 2006. , An effective renement strategy for KNN text classier, Volume 30, Issue 2, (February 2006): Pages 290-298 , doi:10.1016/j.eswa.2005.07.019.

[20] Taeho Jo, 2009. NTC (Neural Text Categorizer): Neural Network for Text Categorization, Volume 2, Issue 2, (28 October 2009).

[21] Zhao Yao and Chen Zhi-Min. 2012. ,An Optimized NBC Approach in Text Classification ,Volume 24, Part C (2012): Pages 1910-1914, DOI: 10.1016/j.phpro.2012.02.281.

[22] Dan R. Herrick and John B . Tyndall. 2013. ,Sustainable Automated Software Deployment Practices , (8 Nov 2013), http://dx.doi.org/10.1145/2504776.2504802.

[23] IEEE Intelligent System, 13(4):18-28, 1998. BibTeX entry. [9], Thorsten Joachims. Text categorization with support vector machines: learning with many relevant

[24] Phil Hayes, software that finds names in text, Intelligent Multimedia on Innovative Applications of Artificial Intelligence, p.49-64, May 01-03, 1990.

[25] Mining text data, CC Aggarwal, CX Zhai, Springer Science & Business Media.

[26] Inductive learning algorithms and representations for text categorization S Dumais, J Platt, D Heckerman, M Sahami, Proceedings of the seventh international conference on Information and

[27] Songbo Tan, Yuefen Wang, Gaowei Wu, Expert Systems with Applications: An International Journal archive, Volume 38 Issue 8, August, 2011, Pages 10264-10273, Pergamon Press, Inc. Tarrytown, NY, USA

[28] A re-examination of text categorization methods. Yiming Yang and Xin Liu. School of Computer Science, Carnegie Mellon University. Pittsburgh, PA 15213-3702