Journal of Engineering and Applied Sciences 14 (7): 2276-2291, 2019
ISSN: 1816-949X

Experimental Comparison of Complexities of Artificial Neural Network and Kolmogorov Distance for News Article Classification

[1]Temitayo M. Fagbola, [2]Ibrahim A. Adeyanju, [3]Ayodele Oloyede, [4]Sijuade Adeyemi,
[2]Olatayo M. Olaniyan and [2]Bolaji A. Omodunbi
[1]Department of Computer Science,
[2]Department of Computer Engineering, Federal University Oye-Ekiti, Ekiti State, Nigeria
[3]Department of Computer Science, Caleb University, Imotan, Nigeria
[4]Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria

Abstract: In today's growing complex multi-domain and multi-language program specifications, the incessant failure of most software products is a great challenge. This is due to the high complexities associated with these products beyond reliable performance tolerable limits which in turn makes their maintenance impossible in an effective manner. On one hand, the inherent complexities of software systems are often ignored at design and implementation stages and on the other hand for such complex software products, maintenance cost becomes very huge in the face of high defect rate. Consequently, evaluating and managing software complexities is key to ensuring that software products are highly understandable, testable, reliable, maintainable, scalable, efficient and cost-effective. In this study, the complexity associated with the use of Artificial Neural Network (ANN) and Kolmogorov Complexity Distance Measure (KCDM) for solving news article classification problem was measured using Lines of Code (LoC) and halstead measure. Similarly, the accuracy and computational efficiency of these classifiers were also determined using true positive rate and testing time measures, respectively. British Broadcasting Corporation (BBC) News dataset composed of 2225 documents corresponding to entertainment, sport, education/technology, politics and business was used for experimental purpose. Based on the results obtained, ANN produced higher classification accuracy at higher complexity and classification time while KCDM yielded lower complexity and testing time but suffers from low classification accuracy. Hence, from a developer's point of view, complexity measurement at design and implementation stages of the software development life cycle is pertinent in other to identify complex designs and codes for refactoring to prevent high software defect rate. From the stakeholders end, effective decisions can be reached by considering the tradeoffs between complexity and accuracy of software products to be used in real-life settings.

Key words: Complexity measurement, Kolmogorov complexity, Halstead measure, neural network, lines of code, news article classification, software

INTRODUCTION

The alarming high defect rate of software applications is of great concern to software stakeholders today. Over 70% of software applications in the market today is developed to meet targeted user's needs in critical application areas, including the banking, health, security and business sectors among others in which services are fully automated. However, software failures in these areas are often too fatal to curtail and usually lead to loss of lives, money and time (Patterson *et al*., 2012). According to Tan (2016), some recent software failures include the $440 million capital loss recorded by Knight capital group on the 1st of August 2012 due to a botched update of computer systems, new year crash of Microsoft Zune on the 31st December, 2008, Excel 2007 arithmetic bug in 2010 air-traffic control system failure in Los Angeles airport which occurred on the 14th day of September, 2004 due to software bug, the Northeast blackout on the 14th day of August, 2003 among others. A project worth $527 million invested by Sainsbury PLC in 2004 was abandoned due to software crisis. In the same vein, a $33.3 million was forfeited by Hudson Bay to software failure in 2005 (Anonymous, 2015).

Corresponding Author: Ibrahim A. Adeyanju, Department of Computer Engineering, Federal University Oye-Ekiti, Ekiti State, Nigeria

Similarly, in a research conducted by National Institute of Standards and Technology (NIST) on software products with high complexity and implications, Chelf and Chou (2008) reported that United State's spends an estimated $59.5 billion annually on unexpected software failures. The report emphasized that software failures could be drastically managed via. robust complexity measurement of source code implementations to realize robust detection and earlier removal of software-associated defects and inherent complexities. During software development life cycle process, software complexity is the degree of difficulty to understand and validate a system design and/or implementation. By implication, software failure is consequent upon negligence to managing complexity of software applications especially at the design and implementation stages. In most cases, software applications become very difficult to maintain due to their associated huge complexities and overrun schedules which is a major cause of their failures (Anonymous, 2015).

Estimation of implementation and software test routines requires that complexity measurement be employed to identify complex designs and codes so as to determine where and what refactoring is expected to improve the quality of the software and prevent high defect rate (Herraiz and Hassan, 2010). This will assist application developers to choose the programming structure and style that can consequently yield optimum program with high reliability. Linders (2014) stated that software complexity is a direct determinant of software costs, reliability and quality. That is the higher the complexity of a software, the lower its quality and the higher the cost needed to manage it (Ogheneovo, 2014). By deduction, systematic code inspection and analysis could help improve the quality of software in any highly competitive and dynamic environment having software products with high volume of lines of code and complexity. A software development process can be controlled via continuous feedbacks provided by complexity measurement (Olabiyisi *et al*., 2014). For example, areas of potential instability in modules of software products can be easily identified during testing and maintenance via complexity measurement (Herraiz and Hassan, 2010). Hence, the complexity measurement can be regarded as a challenging management, engineering and research problem (Olabiyisi *et al*., 2008).

The complexity of a software can be quantified in a number of ways. Among the best-known measures are Lines of Code metrics (LoC) and halstead volume measure. These measures have been validated and extensively used for a number of complexity evaluation tasks (Ramil and Lehman, 2000; Aggarwal *et al*., 2002). However, in text classification problems domain, news article classification is a significant area of interest due to the increasingly growing volume of corpus of news articles on the World Wide Web (WWW). On one hand, news article suffers from a lot of ambiguity during classification due to its various matching categories and the weak reliability indices offered by some classification systems often employed (Biradar and Raikar, 2017; Kaur and Bajaj, 2016) which account for the high complexity associated with such systems. On the other hand, researches in text classification domain that analyzed and evaluated the complexity associated with the underlying source codes of software products to concurrently improve on the quality are difficult to come by.

ANN and Kolmogorov Complexity Distance Measure (KCDM) are among the widely adopted classification systems in high-dimensional problem domains. More specifically, a variety of machine learning systems have adopted ANN and KCDM measure for classification tasks including spam filtering, speech recognition, cellular automata classification, identity and non-parametric testing, malicious URL detection, risk assessment, image recognition, music classification, DNA analysis, radar signal classification, EEG classification, intrinsic plagiarism detection, e-commerce product image and text classification among others (Oyewole and Olugbara, 2017; Revolle *et al*., 2016; Adeyanju *et al*., 2016; Pao *et al*., 2012; Belabbes and Richard, 2008; Omidiora *et al*., 2008; Richard and Doncescu, 2008; Ming and Sleep, 2004). More specifically, ANN has been a good choice for solving text classification problems (Lai *et al*., 2015; Chandra *et al*., 2015; Ng *et al*., 1997; Schutze *et al*., 1995; Wiener *et al*., 1995; Littlestone, 1988).

In this research, the general aim is to establish the significance of complexity measurement of software products during implementation as fundamental and critical towards managing recurring cases of software failures. Experimentally, the evaluation of the LoC and halstead measure complexities of Artificial Neural Network (ANN) and Kolmogorov complexity measure for solving a prototype news article classification problem was conducted. Similarly, the True Positive (TP) and the classification time of both classifiers were also determined. This research emphasizes that inherent code complexity measurement is a mundane software development step and a major performance indicator that defines what software product is reliable, scalable, quality and maintainable beyond just classification accuracy and time efficiency.

Literature review: In this study, software complexity and product quality, trends in code complexity analysis of software applications, trends in multi-label text classification and classification algorithms used in this study are discussed.

Software complexity and product quality: Complexity is an estimate of the resources needed to implement and maintain a software program (Olabiyisi and Adewole, 2008). Precisely it is a measure of the relationship among various elements of the software. With high code complexity, more cost is incurred to manage software programs and at reduced quality. Software products with complex codes often suffer from increasing difficulty to validate, understand and add new features as well as higher defect risk (Linders, 2014). Akanmu *et al*. (2010) stated that the capability of a software developer, size and complexity of a program are the major factors that affect the quality of a software. It is very pertinent to state that complexity is associated with every stage of the Software Development Life Cycle (SDLC) beginning from requirement gathering and specification to implementation (Oram and Wilson, 2011) and this can make software difficult to understand. Most importantly, complexity must be factored out at every stage of the development and kept within tolerable limits. As observed in most applications, compliance with the functional requirements is only ensured at the testing stage with no concern for the associated complexities incurred during the prior stages of development. Herraiz and Hassan (2011) emphasized that most software programs are not reliable and maintainable because the quantitative measurement of complexity of the design and implementation cycles are often ignored. In a related manner, Damasevicius and Stuikys (2010) deduced that when the complexity of a computer program is ignored and it exceeds unknown boundaries, frustration sets in. Computer program becomes too handicapped that maintainability becomes impossible". Therefore, realizing reliable software is borne out of achieving reduced level of complexity, high degree of accuracy and precision when traversing the design stage which is consequent upon efficient overall cost and system efficiency (Thomas, 1979).

Gibbs (1994) pinpointed that lack of software complexity measurement accounts for chronic software crisis. In the same vein, Ho and Basu (2000) advocated the need to evaluate the complexity of classification problems. A number of statistically-based measures that describe the difficulty of classification

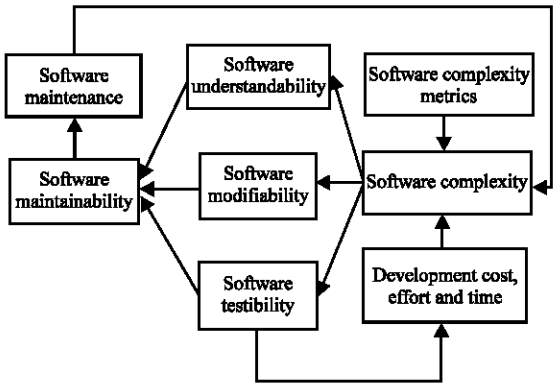

Fig. 1: Block diagram showing the relationship between software product development process and software complexity (Debbarma *et al*., 2013)

problems were discussed in their research. This was done to automatically select the best-fit classifier for some specific classification problems and their associated subproblems. Furthermore, as stated by the seventh law of computer programming, complexity of programs grows steadily until it exceeds the capabilities of the programmer who must maintain it (Dinari, 2015). Consequently, reliability, maintainability, understandability and scalability are the top major challenges of todays' software applications. Guaranteeing that a software under development or already deployed will meet these capabilities is an open problem. As highlighted by Anonymous (2012), there is a strong correlation between software complexity and the quality of software applications. A software program is bad if it has a high level of complexity (Debbarma *et al*., 2013). In Fig. 1, software understandability, modifiability and testability are directly affected by complexity which in turn determines the maintainability and maintenance of the software. This implies that at reduced complexity index, lesser errors will be encountered, software applications will execute faster and become easier to understand, manage, maintain, modify and test.

Trends in code complexity analysis of software applications: A number of researches have been conducted on complexity evaluation of software products. Kinnunen (2006) evaluated the complexities of systems architecture models. The researcher asserted that to commence faster software design with much reduced error, an automatic estimation of the complexity level of such design is a must. A number of complexity measures based on definitions from Object Process Modelling (OPM) were developed. However, states, processes and

objects are the associated entities of OPM. Interface hidden information was evaluated via. the introduction of interface complexity multiplier. The researcher developed models for three distinct systems in the mobile entertainment environment. The essence of the models are to ascertain the suitability of OPM to create an abstraction of the mobile systems and also to provide a parallel evaluation platform to measure complexity. However, the complexity of the mobile entertainment systems models was measured using the newly developed metrics.

In the research of Akanmu *et al.* (2010), the complexities of breadth-first and depth-first search algorithms were evaluated using halstead measure and cyclomatic number. The 4 different implementation languages (C, C++, Pascal and Visual BASIC) were used. The results reported indicate that depth-first search is best implemented in Pascal because with this language, the least program volume, program effort, program difficulty and cyclomatic number is recorded. The same is observed with breadth-first search in which with pascal, the least program volume, program effort, program difficulty and cyclomatic number is recorded. These results indicate that breadth-first search has the least computational complexity when implemented with Pascal. As stated by the authors, there actually exists a trade-off among these metrics as it affects performance; however, software complexity evaluation can help developers to make effective decision on the algorithm that best fits a software specification from a varying set of alternatives.

Furthermore, Olabiyisi *et al.* (2012) carried out a performance evaluation of code-based and procedural cognitive computational complexities on the C-programming language implementations of (10) sorting algorithms (radix, bucket, counting, quick, merge, heap, bubble, shell, insertion and selection). Halstead metrics (program volume, difficulty, effort, reasonable time, level and program length), effective lines of code, procedural cognitive computational complexity, Cyclomatic Complexity (CC) were all computed and determined. The authors concluded that CC could not technically distinguish complexities inherent in different source programs as it yields same final value. Akinwale *et al.* (2012) conducted a comparative analysis of the complexities of Simulated Annealing (SA) and Genetic Algorithm (GA) for proffering solutions to a prototype examination timetabling problem. The complexity of the program source codes were measured using simulation time, Lines of Code (LOC) and Halstead measure which encompassed metrics like program volume, program effort, intelligent content and program level. As deduced from the results presented, GA executes faster than SA. In addition, GA consumes more memory space, requires more coding time and effort than SA (LOC). Both the difficulty and intelligent content of the program are higher in SA than in GA.

In addition, Olabiyisi *et al.* (2013) developed a Software Requirement Specification (SRS) measure to evaluate the complexity of multi-paradigm programming languages. This metric uses the IEEE Software requirement specification standard as a benchmark. The performance of the developed SRS measure was compared with lines of code measure and CC using C++ programming language implementations of 10 sorting algorithms (shell, insertion, selection, radix, quick, merge, heap, bubble, bucket and counting) for evaluation. SRS is claimed as more sensitive than lines of code and CC complexity measures. Furthermore, Honghai and Stefan investigated image complexity using spatial information measure. This was conducted to measure image complexity prior to compression. The work helped to determine the bias between image quality and compression at the implementation stage.

Trends in multi-label text classification: By definition, assigning observations to a class among several predefined classes constitute the major task of classification. By assumption if there exists a classification function $\Phi:O\rightarrow\mathcal{P}(C)\Phi$ and an unknown target function $\breve{\Phi}:O\rightarrow p(C)$, then, multi-label text classification is the process of approximating $\breve{\Phi}$ by means of such Φ that each observation $x\epsilon O$ is assigned to zero or more predefined categories from $C = \{c_1, ..., c_{|c|}\}$ (Skjennum, 2016). Simply put in a prototype classification problem, given a classification function $\gamma:X\rightarrow\mathbb{C}$ where **X** is the document space, $\mathbb{C} = \{C_1, C_2, ..., C_n\}$ is a definite set of classes and is the number of defined classes or categories it is required that γ determines the category, C_i, $i\leq n$ of an unknown test input, X. Generally for any classification problem, an observation, Z can be expressed in the input-output form (x, y) where x is the input into the learning function and y is the output computed using y = f(x). However, given a training dataset, T_D, expressed as:

$$T_D = \{z_n = (x_n, y_n)\}n = 1^N \quad (1)$$

where, $N = |T_D|$ that is the size of T_D, the expected error rate of a hypothesis h is defined as (Meeuwen, 2013):

$$\pi(h) = x \square P_x^E \square h(x) \neq f(x) \square \quad (2)$$

where, the boolean evaluation $[\![h(x)\neq f(x)]\!]$ is 1 if the expression is true and 0 otherwise while P_x is an unknown probability distribution of x. The goal of learning algorithms is to choose h that has low $\pi(h)$. However, to improve on classification accuracy, a learning algorithm can make an attempt to locate an h with the least number of errors in the training data, T_D such that (Li, 2006):

$$e_D(h) \sum_{n=1}^{N} [\![h(x_n) \neq y_n]\!] \quad (3)$$

where, e_D (h) is the hypothesis with the least number of training errors.

News article classification is an example of the multi-label text classification problems. Many researchers have attempted to address this class of problem by developing novel classifiers by conducting evaluation of some existing state-of-the-art classifiers or via. ensemble learning from multiple classifiers to determine the best fit classification function for text documents and articles of some sort. In the research of Kaur and Bajaj (2016), 1000 article subset of BBC News with categories were classified using support vector machine and artificial neural networks. The major categories are entertainment, health, sports and business. Wordnet database was used for synonym identification while snowball was used for stemming. Recall, elapsed time, precision and accuracy were used as performance evaluation metrics. However, the work left no clue to how a number of other vital stages were implemented. These include the neural networks topology used how the data pre-processing was carried out, the process of tokenization and stop-word removal, how the feature extraction and selection were carried out and the various approaches adopted. The results presented were not evaluated and as such not reliable. No attention was paid to the evaluation of the quantitative complexity of the text classification system developed.

Likewise, Zhang *et al.* (2015) developed a character-level convolutional networks for text classification using news dataset. A number of models were adopted including "bag of means", "bag of words", "small and large convolution", "small and large full convolution", ngrams and their variants. However, Sogou News Scopus, AG's News Scopus, DBPedia, Yahoo Answers, Amazon Review Full, Amazon Review Polarity, Yelp Review Full and Yelp Review Polarity are the datasets used for evaluation. Their choice of number of training and testing samples for each of the dataset differs. For example, with Sogou news dataset, 90,000 and 12,000 samples were selected as training and testing datasets, respectively per class and 5 categories were chosen to include "technology", "finance", "sport", "entertainment" and "automobile". They concluded that convolutional network works well for text classification, especially with user-generated dataset without giving any attention to words; there is no machine learning technique or classification system that can work well for all sorts of datasets. However, the quantitative complexity of the character-level convolutional networks developed was not conducted.

In the research of Chan *et al.* (2001), an automated online news categorizor system with personalization was developed to classify channel Asia's financial news. The categorizor has handles for data pre-processing, presentation, storage and classification. There are also handles for registration and webpage retrieval. The system allows general and personalized classification of news with keywords. Support vector machine was employed for classification. The categorizor only supports 10 general categories which are subsets of the reuters-21578 category collections. The categorizor is still in its infancy stage and thus, research is ongoing to make it more robust. Skjennum (2016) explored the characteristics of conceptual, ontological and semantic metadata to develop a scalable, homogeneous news article classification system for global domain based on some extracted IPTC media topic categories and DBpedia entities. The system comprises of an ensemble of classifiers including n-binary multinominal Naive Bayes, feed forward multilayer perceptron network and long short-term memory recurrent neural network. These classifiers were trained with over 1.8 million news articles originating from New York Times annotated corpus. The relevant data and entities were extracted. Categories used include education, health, environment, labour, sport, politics, weather, society, crime, law, art, culture, business among others. A probability of article in more than one category was also determined and reported in percentage.

The implementation, executed on multiple Intel Xeon E5 32 core 2.6 GHz systems, made use of Apache Spark to speed up computation and analysis of the large-scale data. Tools from Spark's library were used to implement deep learning and Naive Bayes as well as for the configuration and training of the ANNs. The performance evaluation metrics used are in two parts: the label and example-based metrics. F-score, micro-and macro-averaged precision, accuracy and recall are the label-based metrics used while subset accuracy and hamming loss are the example-based metrics used. Factors including the length of an article, changes in period of time and inclusion of ontologically related supertypes were identified to affect the quality of news articles classification. However, the question of "how reliable the

Table 1: Summary of some existing works on text classification

Classification problem	Authors and years	Evaluation metrics used
Spam filtering using SVM and kolmogorov complexity	Belabbes and Richard (2008)	Accuracy False positive rate (fall out) True positive rate (recall)
Spam filtering using the kolomgorov complexity analysis	Richard and Doncescu (2008)	Accuracy
Music classification using kolmogorov distance	Zehra *et al.*	Accuracy
Malicious URL detection based on kolmogorov complexity estimation	Pao *et al.* (2012)	False positive, false negative, true positive, true negative and error rate
Intrinsic plagiarism detection using complexity analysis	Leanne and Stan	Recall Precision
Melody classification using a similarity metric based on Kolmogorov complexity	Ming and Sleep (2004)	Accuracy
Document classification for newspaper articles using naive bayes classifier	Dennis and Shreyes	Precision, recall accuracy, F-score
Recurrent convolutional neural networks for text classification	Lai *et al.*	Macro F1, true positive
Web spam classification using supervised artificial neural network algorithms	Chandra *et al.* (2015)	Confusion matrix
Bag-of-concepts document representation for textual news classification	Mourino-Garcia *et al.* (2015)	F1-score

developed multilingual classification system is" pricks the mind. Summary of some recent works on text classification is presented in Table 1. Table 1 highlights the actual classification problem solved and the evaluation metrics used. In all, no code complexity measure was used for evaluation. This often accounts for the intrinsic probable high defect rate that characterizes such solutions.

Classification algorithms: In this study, artificial neural network and Kolmogorov complexity distance measure considered in this study are discussed.

Artificial neural networks: An artificial neuron is the basic building block of Artificial Neural Networks (ANN). In commercial applications, artificial neurons are regarded as processing elements. The major components of an artificial neuron include weighting factors, summation functions, transfer functions, scaling and limiting, output function, error function and back-propagated value and learning functions (Huang, 2009; Dagan *et al.*, 1997; Anderson and McNeill, 1992). Weight is an adaptive coefficient that represents the strength of an input's connection and determines the magnitude of the input signal. A relative weight is associated with each input and required by the summation function. May *et al.* (2011) stressed that the optimal performance of an ANN Model has a functional dependence on the chosen input variables and weight. As such some recommended factors to consider before choosing input variables include training difficulty, relevance, computational effort, comprehensibility and dimensionality. As shown in Fig. 2, the summation function computes the weighted sum of the inputs by summing up the dot product of the components of the input $(i_1, i_2, ..., i)_n$ and their corresponding weights $(w_1, w_2, ..., w_n)$ such that input $1 = i_1 \times w_1$, input $2 = i_2 \times w_2$ and input $n = i_n \times w_n$ in that order. Then, the inputs are summed together and the summation function computes input1+input2+, ..., +inputn.

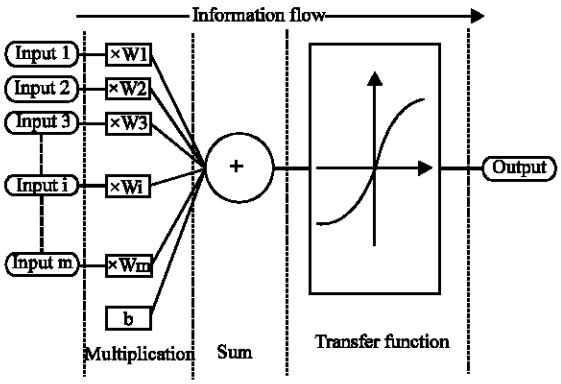

Fig. 2: How an artificial neuron works (May *et al.*, 2011)

According to Anderson and McNeill (1992), some outputs of summation functions are further processed via. an activation function before introduction into the transfer function. This allows the summation output to vary with respect to time. The transfer function transforms the summation output into a working output. A threshold is set and compared with the summation total. If the threshold value is less than or equal to the summation total, a signal is generated else no signal will be generated. Transfer function, though often linear may also be sigmoid (non-linear), hyperbolic tangent, step or sine among others (Krenker *et al.*, 2011). Through the scaling and limiting component of an artificial neuron, a scale factor is multiplied by the transfer value together with an added offset via. the scaling sub-component while the limiting mechanism ensures a restraint of the scaled result within an upper or lower bound. The output function component is responsible for the generation and presentation of the transfer function's result.

Competition can sometimes be incorporated among neighboring processing elements by some network topologies to modify the transfer result (Skjennum, 2016). Suffice it to say that competition determines which

artificial neuron will produce an output. The error function in some learning networks, transforms the difference between the immediate output and the desired output (current error) to match a network architecture or meet a specific purpose. The current error is further propagated backwards into the learning function of another artificial neuron. The value of back-propagation can be a current error or an already-scaled current error. This error term is normally multiplied by the weights of each of the incoming connection for modification prior to the next learning cycle. The learning function, through a learning mode, modifies the weights of the variable connection on the inputs of each artificial neuron via. some neural-oriented algorithms (Huang, 2009). Neural networks are powerful analytical self-learning distributed processors that mimic the structure and functions of the human brain. With ANN, a pool of networks is created and trained to solve specific computational problems.

ANN uses the relationship among inputs and outputs earlier mapped in the training process to determine the system pattern without explicit representation of the physical system and fore-knowledge of the system parameters (Sun *et al*., 2016). Versatility could be achieved in artificial neurons when they are interconnected into ANNs. With this, the network earns flexible learning function and high capability (Krenker *et al*., 2011). However, the complexity of ANN system structure is managed by ensuring that artificial neurons are not interconnected randomly. In this research, ANN was used to learn the inner relationships among tokens and compute a vector space that interprets into a model capable of minimizing the distance among the tokens. The algorithm developed for solving the BBC News classification problem using ANN is presented in Algorithm 1. The term frequencies in the ith document denoted by the vector X_i forms the set of inputs to each unit of the ANN. A set of weights, (w_1, w_2, ..., w_n) denoted as A and associated with each neuron is learnt by computing a function, $pi=A\cdot X_i$, to learn the training data $(X_i,y_i)\forall i\epsilon\{1, ..., n\}$.

Algorithm 1; The algoritm used for solving the BBC News classification problem using ANN:

Inputs: Training Data $(X_i,y_i)\forall i\epsilon\{1, ..., n\}$ and the learning rate, μ
weight vector, A = (w_1, w_2, ..., w_n)
initiliaze A; (set A←0||A← small random numbers
for each BBC news category
repeat
neural network←training data, $T_D = (x_n, y_n)\}_{n=1}^{N}$
if (sign of $A\cdot X_i \neq y_i$) then
update weights in A using the learning rate, μ
else end
until weights in A converge
end for

The learning process was initiated using random weights which are further updated by introducing the current function to the training data. A learning rate was introduced to regulate the magnitude of the update in order to reduce the direction of the neutron's error.

Kolmogorov computational complexity distance measure: Kolmogorov complexity offers the most concise representation of a document/object in a way to completely describe it (Haynes *et al*., 2010). By definition, with a set of formal representation D of an object obj such that D = [S_1, S_2, ..., S_n] where S_1, S_2 are S_n strings that make up D. Let I_i represent the length of string, S_i, in binary bits for all $i\leq n$ which can be re-written as $I_i = |S_i|$. By implication, L = [$|S_1|$, $|S_2|$, ..., S_n|] such that L = [I_1, I_2, ..., I_n]. Simply put, the Kolmogorov complexity of a string is the size of a string S_i with the smallest length I_i such that K = (obj) = min [$|S_1|$, $|S_2|$, ..., S_n|] or min [I_1, I_2, ..., I_n]. However, given any two objects, obj_1 and obj_2, having K complexities K_1 and K_2, respectively,

$$K(obj_2 \mid obj_1)=K(obj_1 c\ obj_2)-K(obj_1) \quad (4)$$

Where:
'c' = The concatenation operator
K (obj_1 c obj_2) = The K complexity
obj_1 and obj_2 = The concatenated

Kolmogorov (1965) with the Kolmogorov distance, the hidden similarities among different set of data can be identified. Hence, the normalized Kolmogorov distance, K_D, between obj_1 and obj_2 is computed as (Cilibrasi and Vitanyi, 2007; Bennett *et al*., 1998):

$$K_D=\left[L(1_1cl_2)-\min\left(L(1_1),L(1_2)\right)\right]/\left[\max\left(L(1_1),L(1_2)\right)\right] \quad (5)$$

That is:

$$K_D(obj_1,obj_2)=\frac{\left(K(obj_1)+K(obj_2)-K(obj_1 c\ obj_2)-\min\left(K(obj_1),K(obj_2)\right)\right)}{\max\left(K(obj_1),K(obj_2)\right)} \quad (6)$$

Simply put:

$$K_D(obj_1,obj_2)=\frac{\left(1-K(obj_1)+K(obj_2)-K(obj_1 c\ obj_2)\right)}{\max\left(K(obj_1),K(obj_2)\right)} \quad (7)$$

MATERIALS AND METHODS

The experimental architecture is presented in 4 stages as shown in Fig. 3:

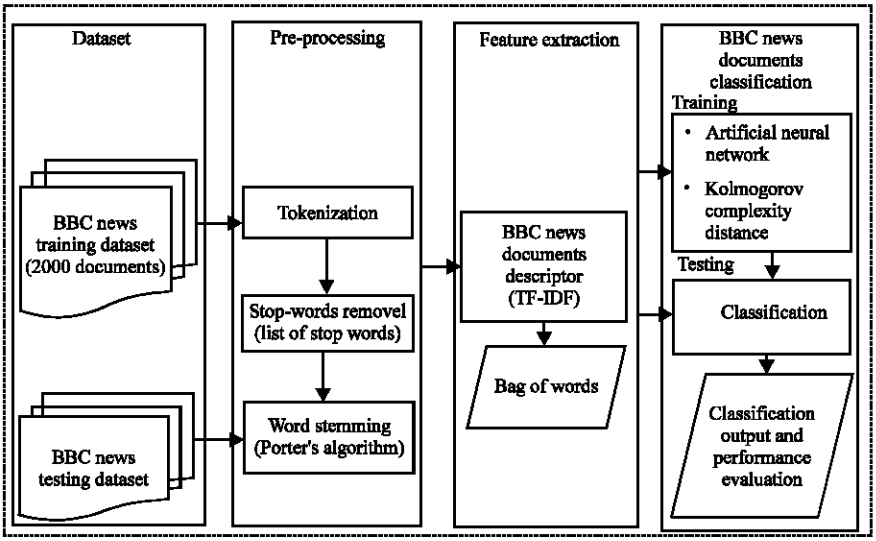

Fig. 3: Our BBC News dataset classification architecture

```
Ad sales boost Time Warner profitQuarterly profits at US media giant TimeWarner jumped 76% to
$1.13bn (£600m) for the three months to December, from $639m year-earlier.The firm, which is now
one of the biggest investors in Google, benefited from sales of high-speed internet connections
and higher advert sales. TimeWarner said fourth quarter sales rose 2% to $11.1bn from $10.9bn.
Its profits were buoyed by one-off gains which offset a profit dip at Warner Bros, and less
users for AOL.Time Warner said on Friday that it now owns 8% of search-engine Google. But its
own internet business, AOL, had has mixed fortunes. It lost 464,000 subscribers in the fourth
quarter profits were lower than in the preceding three quarters. However, the company said AOL's
underlying profit before exceptional items rose 8% on the back of stronger internet advertising
revenues. It hopes to increase subscribers by offering the online service free to TimeWarner
internet customers and will try to sign up AOL's existing customers for high-speed broadband.
TimeWarner also has to restate 2000 and 2003 results following a probe by the US Securities
Exchange Commission (SEC), which is close to concluding.Time Warner's fourth quarter profits
were slightly better than analysts' expectations. But its film division saw profits slump 27% to
$284m, helped by box-office flops Alexander and Catwoman, a sharp contrast to year-earlier, when
the third and final film in the Lord of the Rings trilogy boosted results. For the full-year,
TimeWarner posted a profit of $3.36bn, up 27% from its 2003 performance, while revenues grew
6.4% to $42.09bn. "Our financial performance was strong, meeting or exceeding all of our full-
year objectives and greatly enhancing our flexibility," chairman and chief executive Richard
Parsons said. For 2005, TimeWarner is projecting operating earnings growth of around 5%, and
also expects higher revenue and wider profit margins.TimeWarner is to restate its accounts as
part of efforts to resolve an inquiry into AOL by US market regulators. It has already offered
to pay $300m to settle charges, in a deal that is under review by the SEC. The company said it
was unable to estimate the amount it needed to set aside for legal reserves, which it previously
set at $500m. It intends to adjust the way it accounts for a deal with German music publisher
Bertelsmann's purchase of a stake in AOL Europe, which it had reported as advertising revenue.
```

Fig. 4: Sample of BBC News raw document

- News article dataset collection
- News article dataset pre-processing
- Feature extraction
- Classification using artificial neural network and Kolmogorov complexity distance measure

After the classification stage was completed, the complexities of the underlying source program implementation of ANN and KCDM for solving BBC News classification problem were measured using lines of code and halstead complexity measures.

News article dataset collection: A publicly available research-purpose British Broadcasting Corporation (BBC) News dataset from http://mlg.ucd.ie/datasets/bbc.html was used in this study. A sample of raw BBC News document in the dataset is presented in Fig. 4. The dataset which was prepared by Greene and Cunningham (2006) is composed of 2225 documents corresponding to news on the BBC News website in five topical areas between 2004 and 2005 which are entertainment (386), sport (511), education/technology (401), politics (417) and business (510).

News article dataset pre-processing: Pre-processing of unstructured text data is important to obtaining clean data for reliable classification (Wang and Wang, 2005). Simply put, classification performance is strongly determined by

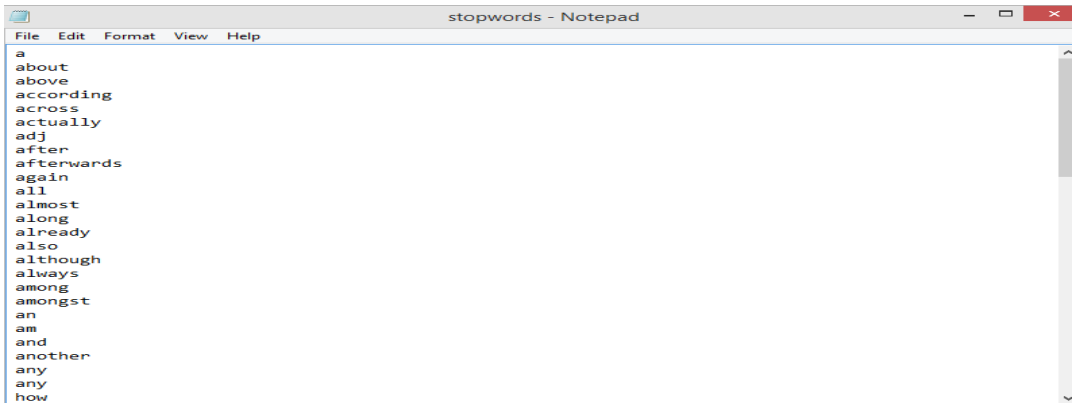

Fig. 5: A portion of stop words list from the BBC News dataset

how appropriately words in text documents have been represented. In this study, tokenization, stop-words removal and stemming were carried out on the BBC News dataset used. Tokenization is basically a process of pruning a set of text or sentences into tokens or words delimited by new line or white spaces (Baharudin *et al*., 2010). In the same vein, term stemming refers to the heuristic approach to morphological derivation of stem/base/root words from each derived/inflected word by removing the affixes. This is a major dimensionality reduction step to manage highly correlated columns (Greene and Cunningham, 2006). In addition, stop words are the frequently occurring group of words that do not add up relevant information to the news classification process (Baharudin *et al*., 2010). These were filtered out from the corpus of terms before final classification. A portion of these stop words for the BBC News dataset is presented in Fig. 5. The dataset was pre-processed by applying porter's algorithm for stemming. Although, Paice/Husk, S and Lovins stemmers exist, the choice of porter's stemmer is borne out of the fact that it is the most efficient and widely adopted approach. Stop words were removed using the BBC News stop word lists and term filtering was conducted to eliminate tokens/words with <3 characters which are often non-descriptive.

Feature extraction: Feature extraction is the process of transforming raw input data into feature sets. Through this process, discriminating features can be extracted from input data to manage dimensionality reduction of the feature space while improving on efficiency (Akinwale *et al*., 2014; Fagbola *et al*., 2017; Temitayo *et al*., 2012). In our feature extraction process, the normalized Term Frequency-Inverse Document Frequency (TF-IDF) scheme was used to extract features from the terms. The choice of TF-IDF was because it performs much better than most other statistical approaches that could be adopted for feature extraction (Joho and Sanderson, 2007) and also because it can be used to represent the weight of terms numerically. TF is the distribution of counts for each term/word in a document and IDF allows for length normalization. The TF-IDF was calculated for each feature and a term-matrix having documents as rows and associated features/TF-IDF scores as columns was generated. The weight for a term i in terms of TF-IDF is given by Mandal and Sen (2014):

$$W_i = \frac{\left(TF_i \times \log\left(\frac{N}{n_i}\right)\right)}{\sqrt{\sum_{i=1}^{n}\left(TF_i \times \log\left(\frac{N}{n_i}\right)\right)^2}} \quad (8)$$

Where:

n_i = The frequency of document of term i
N = The total number of documents

The resultant discriminating features were fed into ANN and Kolmogorov complexity distance measure for similarity index estimation and final classification.

BBC News article dataset classification using artificial neural network and Kolmogorov complexity distance measure: The BBC News discriminating features was splitted into training and testing sets using the k-fold strategy (Mandal and Sen, 2014) with k's value varied between 10 and 70 for the purpose of classification. In Table 2, the category, actual size and the training size used from the BBC News dataset is presented. In the same vein, the total size of the training and testing set used per each k-fold number is presented in Table 3.

Table 2: Category, actual size and training size used from the BBC news dataset

Category	Number of text documents	Number of text documents used
Education and technology	401	350
Sport	511	500
Entertainment	386	300
Business	510	500
Politics	417	350
Total	2225	2000

Table 3: k-fold, training and testing size from the BBC news dataset used

k-fold	Training set for all categories	Number of test document for all categories
10	1800	200
20	1600	400
30	1400	600
40	1200	800
50	1000	1000
60	800	1200
65	700	1300

In all, 2000 news documents out of the total 2225 are used for experimental purpose. The training set is used to represent past observations and obtain learned models that are used for prediction while the testing set is used to represent new observations by being fed into ANN and KCDM discussed in Section 2.4. However, the performance of both algorithms was evaluated using the metrics discussed in the next section.

Performance evaluation metrics: For evaluation purpose, the lines of code and halstead complexity measures were used for source code analysis and complexity measurement. Similarly, number of true positives and the classification time were also determined. However, evaluation of the results obtained was carried out to determine the best-fit algorithm for a prototype news article classification between artificial neural network and Kolmogorov complexity distance measure.

Lines of code metrics: Lines of code is an easy to understand and simple metrics for complexity measurement. It is generally the number of lines of all executable instructions and type declarations (Olabiyisi and Adewole, 2008). There exists a number of variants which are LOCbl, LOCphy, LOCcom and LOCpro. LOCbl is the number of blank lines, LOCphy represents the number of physical lines, LOCcom is represents the number of comment lines while LOCpro is the number of program lines including codes, declarations, definitions and directives (Akinwale *et al*., 2014). A good function length is recommended to be between 4 and 40 lines, file length between 4 and 400 program lines, programs with comments between 30-75% of a file is understandable, below 30% is poorly written and beyond 75% makes it a document (Anonymous, 2012).

Halstead metrics: The major area of interest to halstead complexity measurement include program length (N), vocabulary size (n), program Volume (V), Difficulty level (D), program Level (L), Effort to implement (E), the Time taken to implement (T) and the number of delivered bugs (Akinwale *et al*., 2012; Akanmu *et al*., 2010) Eq. 9:

$$N = N_1 + N_2 \tag{9}$$

where, N_1 and N_2 are the total number of operators and operands contained in a program, respectively. For the vocabulary size (n) Eq. 10:

$$n = n_1 + n_2 \tag{10}$$

where, n_1 and n_2 are the total number of unique operators and operands contained in a program, respectively. The program volume depicts the size of the information contained in a program and is given by Eq. 11:

$$V = N \times \log_2 n \tag{11}$$

As recommended by Anonymous (2012), a function's volume should be between 20 and 1000. However, a volume beyond 1000 means that the function is overloaded. A file's volume should be between 100 and 8000 limits. Difficulty level (D) estimates the degree to which a program is prone to error. Using same operands many times in a program often leads to error. Mathematically Eq. 12:

$$D = \left(\frac{n_1}{2}\right) \times \left(\frac{N_2}{n_2}\right) \tag{12}$$

Program Level (L) determines whether a program is low or high level. A high level program is less susceptible to error than a low level counterpart. By definition Eq. 13:

$$L = \frac{1}{D} \tag{13}$$

The Effort (E) required to understand or implement a program is defined as Eq. 14:

$$E = V \times D \tag{14}$$

The time required to understand or implement a program (T) is given by Eq. 15:

$$T = \frac{E}{18} \tag{15}$$

The number of delivered Bugs (B) depicts the number of errors in a program. It has strong and direct correlation with the overall complexity of the program and is given by Eq. 16:

$$B = \frac{E^{\frac{2}{3}}}{3000} \quad (16)$$

As a recommendation, delivered bugs in a file must not be >2.

True positive: According to Skjennum (2016), the number of correctly assigned documents to a class, category or topic obtained by a classifier represents the true positive value for that classifier. A reliable classifier will normally possess high true positive values.

RESULTS AND DISCUSSION

All the algorithms were implemented using Microsoft Visual C# programming language on a system with Windows 7 64 bit operating system, AMD Athlon (tm) X2 DualCore T3200 central processing unit having a processing speed of 2.2 GHz, 4 GB random access memory and 350 GB hard disk drive.

Table 4: Results of Kolmogorov and ANN code complexity measurement

Metrics	KCDM	ANN
LoC	61	80
n_1	14	25
n_2	16	40
N_1	80	300
N_2	71	144
Program length	151	444
Program vocabulary	30	65
Program difficulty	0.634	45
Program volume	268.478	938.62
Program time (sec)	0.435	39.109

Results on complexity measurement of source implementations of ANN and KCDM: The results of complexity measurement of the source program of KCDM and ANN for solving the BBC News article classification problem is summarized in Table 4 and conceptualized in Fig. 6 using the complexity measures discussed in section 3.5 for evaluation. The results obtained indicate that the total LoC for KCDM and KNN is 61 and 80, respectively. It is worthy of note that the higher the LoC, the more complex the program is (Olabiyisi *et al*., 2012; Akinwale *et al*., 2012). Hence, ANN has higher LoC complexity than KCDM. Similarly, the halstead complexity which is measured by calculating the program length, program vocabulary, program difficulty, program volume and program time was also determined for KCDM and ANN.

The program length of ANN is 444 which is almost (3) Times that of KCDM with a value of 151. Similarly, the program vocabulary of ANN doubles that of KCDM with value 65 and 30, respectively. Simply put, the higher the size of vocabulary, the more complex the program. Furthermore, the program difficulty of ANN obtained is 45. This is approximately 71 times the value of 0.634 obtained for KCDM. By implication it is almost impossible to understand, modify and maintain programs that are highly difficult (Anonymous, 2012). In most practical scenarios, likelihood that a software product will fail increases when such software becomes difficult to understand, modify and maintain. The program volume and time obtained for ANN are 938.62 and 39.109s, respectively. In the same vein, program volume of 268.478 and time of 0.35s were obtained for KSDM. It is also worthy of note that complexity grows with increasing size of program volume and time. Hence, based on the LoC and Halstead evaluations conducted in this study, ANN has a higher complexity than KCDM Table 4.

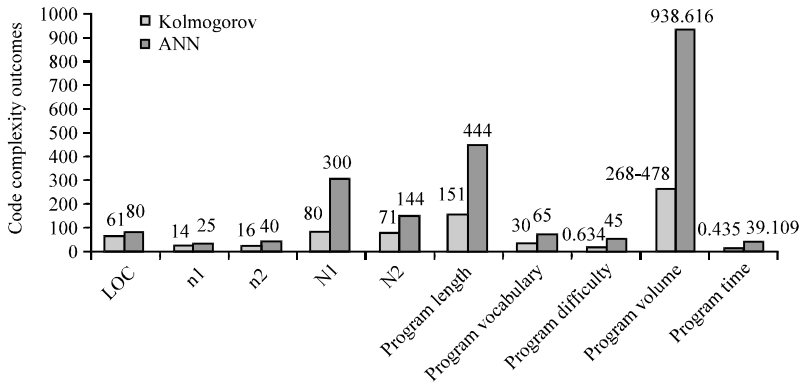

Fig. 6: Graphical representation of the complexity results of KCDM and ANN implementation

Table 5: Testing size, true positive and testing time results for ANN and KCDM from the BBC News dataset used

Number of test document for all categories	True positive (ANN) for all categories	True positive (KCDM) for all categories	Testing time (sec) (ANN) for all categories	Testing time (sec) (KT) for all categories
200	189	168	431	52
400	335	307	571	67
600	537	411	802	96
800	751	628	1025	119
1000	932	663	1429	166
1200	1001	699	1466	179
1300	1026	716	1600	280

Table 6: Classification error for ANN and KCDM

Number of test document for all categories	Error (ANN)	Error (KCDM)
200	11	32
400	65	93
600	63	189
800	49	172
1000	68	337
1200	191	501
1300	274	584

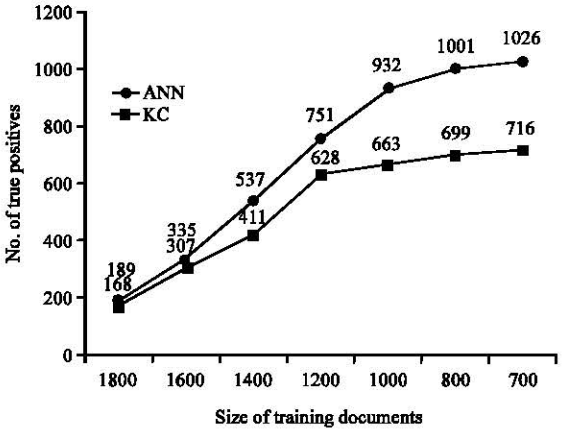

Fig. 7: Plot of number of BBC News training documents and true positives

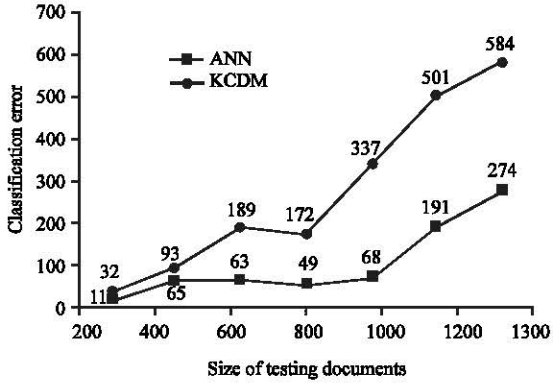

Fig. 8: Graphical representation of the classification error for KCDM and ANN

Results for ANN and KCDM based on true positive and classification time measures: Results were obtained for ANN and KCDM in terms of number of true positives and

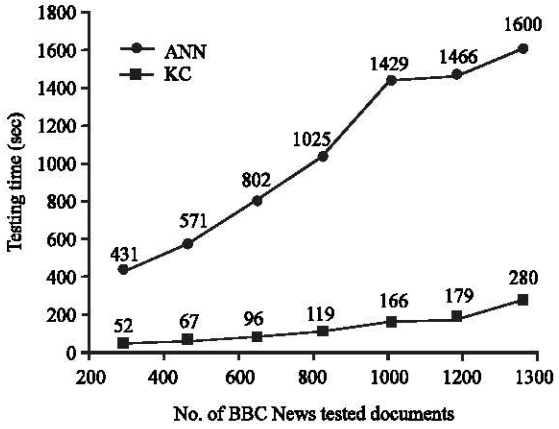

Fig. 9: Number of BBC News testing documents and testing time

classification time as presented in Table 5 in a sequence of 7 different runs. Each run is for a distinct size of training documents spanning from 1800, 1600, 1400, 1200, 1000, 800-700 in that order at k-fold value of 10, 20, 30, 40, 50, 60 and 65, respectively. The plot of the number of true positives obtained by ANN and KCDM against the size of training documents is as shown in Fig. 7. In all the 7 evaluations, ANN produced higher number of true positives than KCDM indicating that it is more accurate. However, the shrink in the size of the training documents widens the error gap produced by ANN and KCDM for the 7 evaluations as presented in Table 6 and Fig. 8 and consequently reduced the accuracy of both algorithms.

This result corroborates with the assertion of (Ayodele *et al*., 2016) that large training size of data is paramount to managing high rate of misclassification and realizing high level of classification accuracy. In the same vein, a plot between the number of BBC News testing documents and testing time is presented in Fig. 9. It is observed that as the number of the testing features increases, the classification time increases correspondingly. Furthermore, KCDM exhibits lower testing time than ANN in all the evaluations conducted.

CONCLUSION

In this study, the complexities, time efficiencies and the accuracy of Artificial Neural Network (ANN) and

Kolmogorov Complexity Distance Measure (KCDM) for solving news article classification problem were evaluated using Lines of Code (LoC), Halstead measure, true positive rate and classification time. Based on the results obtained, ANN performs better than KCDM in terms of accuracy only. On the other hand, KCDM produced lower complexities and time efficiencies than ANN in all the evaluations conducted. This study strongly recommends code complexity analysis as a core part of the test routines during design and implementation stages of software development to ensure high reliability and huge post-deployment maintenance cost savings. Furthermore, it will help to identify complex designs and codes that can be refactored to reduce associated complexities. This is more important in critical environments with zero-tolerance to software defects. The major findings of our experiments are summarized thus:

On large and well-structured training sets, ANN is more superior than KCDM to realizing high classification accuracy. It produces higher number of True positives than KCDM in all the experiments.

As the size of the testing set increases, ANN suffers from higher testing time than KCDM. With testing set containing 1300 features, the testing time of ANN is approximately 8 times that of KCDM.

From the experiments, based on LoC and Halstead metrics, KCDM produces the best complexity result but ANN incurs huge time and code complexities which calls for further investigation.

RECOMMENDATIONS

Future researches may investigate the code complexities of some other commonly used classifiers like SVM, KNN and Naive Bayes on news article classification. Similarly, the evaluation can be extended by introducing other complexity evaluation measures like McCabe and Cyclomatic complexity measures.

REFERENCES

Adeyanju, I.A., O.O. Awodoye and E.O. Omidiora, 2016. Performance evaluation of an improved self-organizing feature map and modified counter propagation network in face recognition. Br. J. Math. Comput. Sci., 14: 1-12.

Aggarwal, K.K., Y. Singh and J.K. Chhabra, 2002. An integrated measure of software maintainability. Proceedings of the 2002 Annual Symposium on Reliability and Maintainability, January 28-31, 2002, IEEE, Seattle, Washington, pp: 235-241.

Akanmu, T.A., S.O. Olabiyisi, E.O. Omidiora, C.A. Oyeleye and M.A. Mabayoje *et al*., 2010. Comparative study of complexities of breadth-first search and depth-first search algorithms using software complexity measures. Proceedings of the 2010 World Congress on Engineering (WCE'10), June 30-July 2, 2010, London, England, UK., ISBN:978-988-17012-9-9, pp: 1-6.

Akinwale, C., S. Olatunde, E. Olusayo and J. Babalola, 2012. Performance evaluation of simulated annealing and genetic algorithm in solving examination timetabling problem. Sci. Res. Essays, 7: 1727-1733.

Akinwale, O.C., O.S. Olatunde, O.E. Olusayo and F. Temitayo, 2014. Hybrid metaheuristic simulated annealing and genetic algorithm for solving examination timetabling problem. Intl. J. Comput. Sci. Eng., 3: 7-22.

Anderson, D. and G. McNeill, 1992. Artificial neural networks technology. MSc Thesis, Data & Analysis Center for Software (DACS), New York, USA.

Anonymous, 2012. Code complexity-software complexity. Verifysoft Technology GmbH, Germany. https://www.verifysoft.com/en-code-complexity.html

Anonymous, 2015. Causes of software crisis. UK Essays, Nottingham, England, UK. https://www.ukessays.com/essays/information-technology/symptoms-and-primary-causes-of-software-crisis-information-technology-essay.php

Ayodele, O., F. Temitayo, S.O. Olabiyisi, E.O. Omidiora and J. Oladosu, 2016. Development of a modified local binary pattern-gabor wavelet transform aging invariant face recognition system. Proceedings of the International Conference on Computing Research and Innovations (OcRI'16), September 7-9, 2016, University of Ibadan, Ibadan, Nigeria, pp: 108-114.

Baharudin, B., L.H. Lee and K. Khan, 2010. A review of machine learning algorithms for text-documents classification. J. Adv. Inform. Technol., 1: 4-20.

Belabbes, S. and G. Richard, 2008. On using SVM and Kolmogorov complexity for spam filtering. Proceedings of the 21st International Conference on Florida Artificial Intelligence Research Society (FLAIRS'08), May 15-17, 2008, Association for the Advancement of Artificial Intelligence, Menlo Park, California, USA., pp: 130-135.

Bennett, C.H., P. Gacs, M. Li, P.M. Vitanyi and W.H. Zurek, 1998. Information distance. IEEE. Trans. Inf. Theor., 44: 1407-1423.

Biradar, S. and M. Raikar, 2017. Performance analysis of text classifiers based on news articles-a survey. Indian J. Sci. Res., 15: 156-161.

Chan, C.H., A. Sun and E.P. Lim, 2001. Automated online news classification with personalization. Proceedings of the 4th International Conference on Asian Digital Library (ICADL'01), December 10-12, 2001, Bangalore, India, pp: 320-329.

Chandra, A., M. Suaib and D. Beg, 2015. Web spam classification using supervised artificial neural network algorithms. Adv. Comput. Intell. Intl. J., 2: 21-30.

Chelf, B. and A. Chou, 2008. Controlling software complexity: The business case for static source code analysis. Technical report, Coverity, San Francisco, California, USA. https://www.bitpipe.com/detail/RES/1204657851-780.html

Cilibrasi, R.L. and P.M.B. Vitanyi, 2007. The Google similarity distance. IEEE Trans. Knowl. Data Eng., 19: 370-383.

Dagan, I., Y. Karov and D. Roth, 1997. Mistake-driven learning in text categorization. Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing (EMNLP'97), August 1-2, 1997, ACL, Providence, Rhode Island, USA., pp: 1-9.

Damasevicius, R. and V. Stuikys, 2010. Metrics for evaluation of metaprogram complexity. Comput. Sci. Inf. Syst., 7: 769-787.

Debbarma, M.K., S. Debbarma, N. Debbarma, K. Chakma and A. Jamatia, 2013. A review and analysis of software complexity metrics in structural testing. Intl. J. Comput. Commun. Eng., 2: 129-133.

Dinari, F., 2015. Halstead complexity metrics in software engineering. J. Renewable Nat. Resour. Bhutan, 3: 418-424.

Fagbola, T.M., S.O. Olabiyisi, F.I. Egbetola and A. Oloyede, 2017. Review of technical approaches to face recognition with varying pose and illumination in unconstrained scenes. FUOYE. J. Eng. Technol., 2: 1-8.

Greene, D. and P. Cunningham, 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. Proceedings of the 23rd International Conference on Machine Learning, June 25-29, 2006, ACM, Pittsburgh, Pennsylvania, USA., ISBN:1-59593-383-2, pp: 377-384.

Haynes, K., R. Kulkarni, R. Stough and L. Schintler, 2010. Exploring region classifier based on Kolmogorov complexity. MSc Thesis, GMU School of Public, George Mason University, Fairfax, Virginia.

Herraiz, I. and A.E. Hassan, 2010. Beyond Lines of Code: Do We Need More Complexity Metrics?. In: Making Software: What Really Works and Why We Believe it, Oram, A. and G. Wilson (Eds.). O'Reilly Media, Sebastopol, California, USA., ISBN:978-0-596-80832-7, pp: 125-141.

Ho, T.K. and M. Basu, 2000. Measuring the complexity of classification problems. Proceedings of the 15th International Conference on Pattern Recognition Vol. 2, September 3-7, 2000, IEEE, Barcelona, Spain, pp: 43-47.

Huang, Y., 2009. Advances in artificial neural networks-methodological development and application. Algorithms, 2: 973-1007.

Joho, H. and M. Sanderson, 2007. Document frequency and term specificity. Proceedings of the Conference on Large Scale Semantic Access to Content (Text, Image, Video and Sound), May 30-June 01, 2007, ACM, Pittsburgh, Pennsylvania, USA., pp: 350-359.

Kaur G. and K. Bajaj, 2016. News classification and its techniques: A review. IOSR. J. Comput. Eng., 18: 22-26.

Kinnunen, M.J., 2006. Complexity measures for system architecture models. Ph.D Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

Kolmogorov, A.N., 1965. Three approaches to the quantitative definition ofinformation. Problems Inform. Transmission, 1: 1-7.

Krenker, A., J. Bester and A. Kos, 2011. Introduction to the Artificial Neural Networks. In: Artificial Neural Networks-Methodological Advances and Biomedical Applications, Suzuki, K. (Ed.). InTech, Rijeka, Croatia, ISBN: 978-953-307-243-2, pp: 3-18.

Lai, S., L. Xu, K. Liu and J. Zhao, 2015. Recurrent convolutional neural networks for text classification. Proceedings of the 29th International Conference on Artificial Intelligence Vol. 333, January 25-30, 2015, Association for the Advancement of Artificial Intelligence, Menlo Park, California, USA., pp: 2267-2273.

Li, L., 2006. Data complexity in machine learning and novel classification algorithms. Ph.D Thesis, California Institute of Technology, California, USA.

Linders, B., 2014. Using complexity measurements to improve software quality. InfoQ., London, England, UK. https://www.infoq.com/news/2014/10/complexity-software-quality

Littlestone, N., 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Mach. Learn., 2: 285-318.

Mandal, A.K. and R. Sen, 2014. Supervised learning methods for bangla web document categorization. Intl. J. Artif. Intell. Appl., 5: 93-105.

May, R., G. Dandy and H. Maier, 2011. Review of Input Variable Selection Methods for Artificial Neural Networks. In: Artificial Neural Networks-Methodological Advances and Biomedical Applications, Suzuki, K. (Ed.). InTech, Rijeka, Croatia, ISBN:978-953-307-243-2, pp: 19-45.

Meeuwen, F.V., 2013. Multi-label text classification of news articles for ASDMedia. MSc Thesis, Department of Information and Computing Sciences, Utrecht University, Utrecht, Netherlands.

Ming, L. and R. Sleep, 2004. Melody classification using a similarity metric based on Kolmogorov complexity. Sound Music Comput., 1: 1-5.

Mourino-Garcia, M., R. Perez-Rodriguez and L.E. Anido-Rifon, 2015. Bag-of-concepts document representation for textual news classification. Intl. J. Comput. Ling. Appl., 6: 173-188.

Ng, H.T., W.B. Goh and K.L. Low, 1997. Feature selection, perceptron learning and a usability case study for text categorization. Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Philadelphia, Pennsylvania, United States, July 27-31, 1997, ACM, New York, USA., pp: 67-73.

Ogheneovo, E.E., 2014. On the relationship between software complexity and maintenance costs. J. Comput. Commun., 2: 1-16.

Olabiyisi S.O. and O.A. Adewole, 2008. On the software complexity measures of bubble sort algorithm. Res. J. Appl. Sci., 3: 5-9.

Olabiyisi S.O., O.O. Elijah and O.I. Esther, 2012. Performance evaluation of procedural cognitive complexity metric and other code based complexity metrics. Intl. J. Sci. Eng. Res., 3: 1-7.

Olabiyisi, S.O., A.B. Adetunji and T.R. Olusi, 2013. Using software requirement specification as complexity metric for multi-paradigm programming languages. Intl. J. Emerging Technol. Adv. Eng., 3: 562-569.

Olabiyisi, S.O., E.O. Omidiora and M.O. Balogun, 2014. A complexity metric for multi-paradigm programming languages. Intl. J. Emerging Technol. Adv. Eng., 4: 59-65.

Olabiyisi, S.O., R.O. Ayeni, E.O. Omidiora and O.A. Bello, 2008. Universal scan machine for complexity measurement of computer programs. J. Comput. Applic., 15: 73-81.

Omidiora, E.O., A.O. Fakolujo, R.O. Ayeni, I.A. Adeyanju and E.O. Oyetunji et al., 2008. Optimised fisher discriminant analysis for recognition of faces having black features. J. Eng. Appl. Sci., 3: 524-531.

Oyewole, S.A. and O.O. Olugbara, 2017. Product image classification using Eigen Colour feature with ensemble machine learning. Egypt. Inf. J., 30: 1-18.

Pao, H.K., Y.L. Chou and Y.J. Lee, 2012. Malicious URL detection based on Kolmogorov complexity estimation. Proceedings of the 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT'12) Vol. 1, December 4-7, 2012, IEEE, Macau, China, ISBN:978-1-4673-6057-9, pp: 380-387.

Patterson, S., J. Strasburg and J. Bunge, 2012. Knight upgrade triggered old trading system, big losses. The Wall Street Journal Newspaper, New York, USA. https://www.wsj.com/articles/SB10000872396390444318104577589694289838100

Ramil, J.F. and M.M. Lehman, 2000. Metrics of software evolution as effort predictors-a case study. Proceedings of the 2000 International Conference on Software Maintenance (ICSM), October 11-14, 2000, IEEE, San Jose, California, pp: 163-172.

Revolle, M., N. Le Bihan and F. Cayre, 2016. Algorithmic Information Theory for Automatic Classification. University of Grenoble, Grenoble, France.

Richard, G. and A. Doncescu, 2008. Spam filtering using Kolmogorov complexity analysis. Intl. J. Web Grid Serv., 4: 136-148.

Schutze, H., D.A. Hull and J.O. Pedersen, 1995. A comparison of classifiers and document representations for the routing problem. Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 09-13, 1995, ACM, Seattle, Washington, USA., ISBN:0-89791-714-6, pp: 229-237.

Skjennum, P.L., 2016. Multilingual news article classification. Master's Thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway.

Sun, Y., D. Wendi, D.E. Kim and S.Y. Liong, 2016. Application of artificial neural networks in groundwater table forecasting-a case study in a Singapore swamp forest. Hydrology Earth Syst. Sci., 20: 1405-1412.

Tan, G., 2016. A collection of well-known software failures. Ph.D Thesis, Pennsylvania State University, Pennsylvania, USA.

Temitayo, F., O. Stephen and A. Abimbola, 2012. Hybrid GA-SVM for efficient feature selection in E-mail classification. Comput. Eng. Intell. Syst., 3: 17-28.

Wang, Y. and X.J. Wang, 2005. A new approach to feature selection in text classification. Proceedings of the 2005 International Conference on Machine Learning and Cybernetics Vol. 6, August 18-21, 2005, IEEE, Guangzhou, China, pp: 3814-3819.

Wiener, E., J.O. Pedersen and A.S. Weigend, 1995. A neural network approach to topic spotting. Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval, April 24-26, 1995, Las Vegas, Nevada, pp: 317-332.

Zhang, X., J. Zhao and Y. LeCun, 2015. Character-Level Convolutional Networks for Text Classification. In: Advances in Neural Information Processing Systems, Cortes, C., N.D. Lawrence, D.D. Lee, M. Sugiyama and R. Garnett (Eds.). Curran Associates, Inc., New York, USA., pp: 649-657.