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

1Temitayo M. Fagbola, 2Ibrahim A. Adeyanju, 3Ayodele Oloyede, 4Sijuade Adeyemi, 3Olatayo M. Olaniyi and 2Bolaji A. Omodunbi
1Department of Computer Science, 2Department of Computer Engineering, Federal University Oye-Ekiti, Ekiti State, Nigeria 3Department of Computer Science, Caleb University, Imotan, Nigeria 4Department 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 (KCMD) 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 KCMD 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, Cheif 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 (Herruz 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 (Oghenevevo, 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 (Olabiyis 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 (Herruz and Hassan, 2010). Hence, the complexity measurement can be regarded as a challenging management, engineering and research problem (Olabiyis 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 Donecsco, 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 (Olabiysi 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 (Lindens, 2014). Akamu 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. Hereraiz 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 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 today’s 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 (Debbarna 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

![Fig. 1: Block diagram showing the relationship between software product development process and software complexity (Debbarna et al., 2013)](image-url)
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 Akamnu 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, Olabiysi 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, Olabiysi 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, 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 image. 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 $\Phi: \mathbb{O} \rightarrow \mathbb{P}(\mathbb{C})$ and an unknown target function $\Phi: \mathbb{O} \rightarrow \mathbb{P}(\mathbb{C})$, then, multi-label text classification is the process of approximating $\Phi$ by means of such $\Phi$, each observation $x \in \mathbb{O}$ is assigned to zero or more predefined categories from $\mathbb{C} = \{c_1, ..., c_n\}$ (Skjennud, 2016). Simply put in a prototype classification problem, given a classification function $\gamma: \mathbb{X} \times \mathbb{C}$ where $\mathbb{X}$ is the document space, $\mathbb{C} = \{c_1, c_2, ..., c_m\}$ is a definite set of classes and is the number of defined classes or categories it is required that $\gamma$ determines the category, $c_i \in \mathbb{I}$ 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, $T_p$, expressed as:

$$T_p = \{(x, y) \in \mathbb{X} \times \mathbb{Y} | n = 1\}$$

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

$$L(h) = x \in \mathbb{X} \sum_{h_e \neq y} p_e \cdot \mathbb{I}$$

(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, \( T_e \), such that (Li, 2006):

\[
e_c(h) \sum_{i=1}^{F} \delta(h(x_i) \neq y_i) \tag{3}
\]

where, \( e_c(h) \) 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 categorizer system with personalization was developed to classify channel Asia’s financial news. The categorizer 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 categorizer only supports 10 general categories which are subsets of the reuters-21578 category collectors. The categorizer is still in its infancy stage and thus, research is ongoing to make it more robust. Skjemaa (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 multinomial Naïve 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 Naïve 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
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 input 1 + input 2 + ..., + input n.

```
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 (Skjemnum, 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 the 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, \( i 
Rightarrow A \cdot X_i \), to learn the training data \( (X_i, y_i) \forall i \in \{1, ..., n\} \).

**Algorithm 1:** The algorithm used for solving the BBC News classification problem using ANN:

Inputs: Training Data \((X_i, y_i)\forall i \in \{1, ..., n\}\) and the learning rate, \( \mu \)

1. Initialize \( \lambda_i \) (set \( A \cdot 0 \lambda \) - small random numbers for each BBC news category)
2. Repeat
   - Neural network - training data, \( T_0 = (x_0, y_0) \)
   - If (sign of \( A \cdot X_i + y_i \) then
     - Update weights in \( A \) using the learning rate, \( \mu \)
   - Else end
3. Until weights in \( A \) converge

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 neuron'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 \( S_i \) are \( S_0 \) strings that make up \( D \). Let \( I_0 \) represent the length of string \( S_i \) in binary bits for all \( i \in n \) which can be re-written as \( I_0 = [S_i] \). By implication, \( I_0 = [S_1, [S_2, ..., S_n]] \) such that \( L = [I_0, I_1, ..., I_n] \). Simply put, the Kolmogorov complexity of a string is the size of a string \( S_i \) with the smallest length \( I_0 \) such that \( K = \text{obj} = \text{min}[S_1, [S_2, ..., S_n]] \) or \( \text{min}[I_0, I_1, ..., I_n] \). However, given any two objects, \( obj_1 \) and \( obj_2 \) having \( K \) complexities \( K_{1} \) and \( K_{2} \), respectively,

\[
K(obj_1 \mid obj_2) = K(obj_1 \circ obj_2) - K(obj_2)
\]  

(4)

Where:

- \( c \) = The concatenation operator
- \( K(obj_1 \circ obj_2) = The\, K\, complexity\) of \( obj_1 \) 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, \( K_n \), between \( obj_1 \) and \( obj_2 \) is computed as (Cilibrasi and Vitanyi, 2007; Bennett et al., 1998):

\[
K_n = \left[ \frac{L(1, e_1)}{\text{min}(L(1), L(e_1))} \right] / \left[ \frac{\text{max}(L(1), L(e_1))}{} \right]
\]

(5)

That is:

\[
K_n(obj_1, obj_2) = \frac{K(obj_1) + K(obj_2) - K(obj_1 \circ obj_2)}{\text{min}(K(obj_1), K(obj_2))}
\]

(6)

Simply put:

\[
K_n(obj_1, obj_2) = \frac{1 - K(obj_1) + K(obj_2) - K(obj_1 \circ obj_2)}{\text{max}(K(obj_1), K(obj_2))}
\]

(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

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
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):

\[
W_i = \sqrt{\sum_{j=1}^{n} \left( \frac{TF_j \times \log \left( \frac{N}{n_i} \right)}{n} \right)^2}
\]

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 split 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

<table>
<thead>
<tr>
<th>Category</th>
<th>Number of text documents</th>
<th>Number of text documents used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Education and technology</td>
<td>401</td>
<td>350</td>
</tr>
<tr>
<td>Sport</td>
<td>511</td>
<td>500</td>
</tr>
<tr>
<td>Entertainment</td>
<td>386</td>
<td>300</td>
</tr>
<tr>
<td>Business</td>
<td>510</td>
<td>500</td>
</tr>
<tr>
<td>Politics</td>
<td>417</td>
<td>350</td>
</tr>
<tr>
<td>Total</td>
<td>2225</td>
<td>2000</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>k-fold</th>
<th>Training set for all categories</th>
<th>Number of test documents for all categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1800</td>
<td>200</td>
</tr>
<tr>
<td>20</td>
<td>1600</td>
<td>400</td>
</tr>
<tr>
<td>30</td>
<td>1400</td>
<td>600</td>
</tr>
<tr>
<td>40</td>
<td>1200</td>
<td>800</td>
</tr>
<tr>
<td>50</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>60</td>
<td>800</td>
<td>1200</td>
</tr>
<tr>
<td>65</td>
<td>700</td>
<td>1300</td>
</tr>
</tbody>
</table>

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 (Olabiysi 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 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; Akamnu et al., 2010) Eq. 9:

\[ N = N_1 + N_2 \]  \hspace{1cm} (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:

\[ n = n_1 + n_2 \]  \hspace{1cm} (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:

\[ V = N \times \log_2 n \]  \hspace{1cm} (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:

\[ D = \left( \frac{n_1}{2} \right) \times \left( \frac{N_2}{n_1} \right) \]  \hspace{1cm} (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:

\[ L = \frac{1}{D} \]  \hspace{1cm} (13)

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

\[ E = V \times D \]  \hspace{1cm} (14)

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

\[ T = \frac{E}{18} \]  \hspace{1cm} (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:

\[ B = \frac{1}{3000} \frac{E^2}{I} \]  

(16)

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

**True positive:** According to Skjennud (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>
<thead>
<tr>
<th>Metrics</th>
<th>KCDM</th>
<th>ANN</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoC</td>
<td>61</td>
<td>80</td>
</tr>
<tr>
<td>n1</td>
<td>14</td>
<td>25</td>
</tr>
<tr>
<td>n2</td>
<td>16</td>
<td>40</td>
</tr>
<tr>
<td>N1</td>
<td>80</td>
<td>300</td>
</tr>
<tr>
<td>N2</td>
<td>71</td>
<td>144</td>
</tr>
<tr>
<td>Program length</td>
<td>151</td>
<td>444</td>
</tr>
<tr>
<td>Program vocabulary</td>
<td>30</td>
<td>65</td>
</tr>
<tr>
<td>Program difficulty</td>
<td>0.634</td>
<td>45</td>
</tr>
<tr>
<td>Program volume</td>
<td>268.478</td>
<td>938.62</td>
</tr>
<tr>
<td>Program time (sec)</td>
<td>0.435</td>
<td>39.109</td>
</tr>
</tbody>
</table>

**Results on complexity measurement 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 (Olabyisie 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.

![Graphical representation of the complexity results of KCDM and ANN implementation](image)

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

<table>
<thead>
<tr>
<th>Number of test document for all categories</th>
<th>True positive (ANN) for all categories</th>
<th>True positive (KCDM) for all categories</th>
<th>Testing time (sec) (ANN) for all categories</th>
<th>Testing time (sec) (KCDM) for all categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>189</td>
<td>168</td>
<td>431</td>
<td>52</td>
</tr>
<tr>
<td>400</td>
<td>335</td>
<td>307</td>
<td>571</td>
<td>67</td>
</tr>
<tr>
<td>600</td>
<td>537</td>
<td>411</td>
<td>802</td>
<td>96</td>
</tr>
<tr>
<td>800</td>
<td>751</td>
<td>628</td>
<td>1025</td>
<td>119</td>
</tr>
<tr>
<td>1000</td>
<td>982</td>
<td>663</td>
<td>1429</td>
<td>166</td>
</tr>
<tr>
<td>1200</td>
<td>1001</td>
<td>699</td>
<td>1466</td>
<td>179</td>
</tr>
<tr>
<td>1500</td>
<td>1026</td>
<td>716</td>
<td>1600</td>
<td>280</td>
</tr>
</tbody>
</table>

Table 6: Classification error for ANN and KCDM

<table>
<thead>
<tr>
<th>Number of test document for all categories</th>
<th>Error (ANN)</th>
<th>Error (KCDM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>11</td>
<td>32</td>
</tr>
<tr>
<td>400</td>
<td>65</td>
<td>93</td>
</tr>
<tr>
<td>600</td>
<td>63</td>
<td>189</td>
</tr>
<tr>
<td>800</td>
<td>49</td>
<td>172</td>
</tr>
<tr>
<td>1000</td>
<td>68</td>
<td>337</td>
</tr>
<tr>
<td>1200</td>
<td>191</td>
<td>501</td>
</tr>
<tr>
<td>1500</td>
<td>274</td>
<td>584</td>
</tr>
</tbody>
</table>

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 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.

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

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 Naïve Bayes on news article classification. Similarly, the evaluation can be extended by introducing other complexity evaluation measures like McCabe and Cyclomatic complexity measures.

REFERENCES


