# A Noun-Centric Keyphrase Extraction Model: Graph-Based Approach

Rilwan O. Abimbola
First Technical University, Ibadan, Nigeria
Email: rilwan.abimbola@tech-u.edu.ng

Iyabo O. Awoyelu, Folasade O. Hunsu, Bodunde O. Akinyemi, and Ganiyu A. Aderounmu
Obafemi Awolowo University, Ile-Ife, Nigeria
Email: {iawoyelu, fohunsu, bakinyemi, gaderoun}@oauife.edu.ng

***Abstract*—The graph-based approach has proven to be the most effective method of extracting keyphrases. Existing graph-based extraction methods do not include nouns as a component, resulting in keyphrases that are not noun-centric, leading to low-quality keyphrases. Also, the clustering approach employed in most of the keyphrase extraction has not yielded good results. This study proposed an improved model for extracting keyphrases that uses a graph-based model with noun phrase identifiers and effective clustering techniques. Relevant data was collected from selected documents in the English language. A graph-based model was formulated by integrating the textrank algorithm for node ranking, a noun phrase identifier for noun phrase scoring, an affinity propagation algorithm for selecting cluster groups, and k-means for clustering. The formulated model was implemented and evaluated by benchmarking it with an existing model using recall, f-measure, and precision as performance metrics. Final results showed that the developed model has a higher precision of 5.5%, a recall of 5.3%, and an f-measure score of 5.5% over the existing model. This implied that the noun-centric keyphrase extraction ensured high-quality keyphrase extraction.**

***Index Terms*—keyphrase, keyphrase extraction, noun-centric, graph-based model, clustering**

## I. Introduction

A keyphrase is an important phrase or single word that connects the features of a document. For example, a document about "building a house" should contain keyphrases such as "foundation" or "roofing". A keyphrase provides a top-level summary of a document and is useful in various applications of text mining, such as document categorization, clustering, and classification. Prior to the advent of automatic keyphrase extraction, the task of keyphrase extraction was usually performed manually by humans, which made the process time-consuming. With the growth of the Internet, the scale of information is expanding, and thus the manual process of annotating documents with keyphrases is unproductive for expert human indexers.

Manuscript received December 3, 2021; revised May 26, 2022; accepted June 27, 2022.

Therefore, the use of text data such as articles, news, and scientific publications around the world has made it critical to develop an effective and efficient method for automatically identifying quality keyphrases. Keyphrases are useful for text clustering [1], text classification [2], and document summarization [3]. Keyphrases are concise summaries that provide a description of a document. They help document readers understand the content of a document, and they are usually chosen by the document's author or professional indexers. Keyphrases are essentially a shortlist of phrases (usually five to fifteen words, which can be noun phrases) describing various areas reviewed in a specific document [4]. Keyphrases are important in determining the context of a document. Keyphrase extraction eliminates the need to read through a document to determine if it is relevant to one's search. The automation of this process will save a great deal of time and money compared to hiring a human indexer. Keyphrase extraction is a step towards solving some text mining problems, such as text summarization, headline generation, and automatic essay grading. Keyphrases could facilitate skimming of a text when the keyphrases are highlighted [5]. They can also be used for clustering, back of book indexing, searching, and document classification. Keyphrases also aid in the refinement of user queries by search engines.

A graph model is a representation of a real-life abstraction that employs graph theory to denote an abstraction with edges and nodes that allow for further investigation of the real-life abstraction. Graph-based methods first create a word graph, centered on the document's word co-occurrences, and then rank the words based on their scores. Subsequently, the top-ranked words are the important keyphrases.

Because manually assigning keys is a difficult task, various machine learning approaches have been proposed. Some approaches are unsupervised while others are supervised, and some work uses a combination of the two. Only a few are available for free, making it difficult to reproduce prior results or employ key concepts in alternative applications such as question-answering systems, text summarization, headline generation, and information retrieval, to name a few. In recent years, the

doi: 10.12720/jait.13.6.578-589

graph-based approach has been the most successful method of extracting keyphrases [6]. This is because the graph model examines the relationship between two words or phrases as well as their relationships with other words or phrases throughout the document.

It was noted, however, that existing graph-based methods for key extraction did not consider nouns as part of them, resulting in a key that may or may not contain a noun. Existing graph-based models that have used the k-means algorithm have to set a predefined number of clusters before extraction. The k-means algorithm used in previous works had the disadvantage of requiring users to select the number of clusters before the extraction process began. There is a need for an improved model that employs a graph-based model with noun phrase identifier in extracting keyphrases and a clustering algorithm that automatically selects the number of clusters. Therefore, this study aims to address these shortcomings of keyphrase extraction by introducing the noun phrase identifier, as most keyphrases are nouns or contain nouns [7].

## II. Related Works

Traditional approaches to automatic keyphrase selection rely on assigning or extracting keyphrases, both employing machine learning techniques. Keyphrase assignment chooses keyphrases out of a pre-existing restricted list of phrases and assigns them to a sentence that accurately describes them. Keyphrase extraction, contrarily, obtains these terms directly out of the text. Specifically, the words and phrases have to be available in the subjects of the text. Several methodologies attempt to define what a keyphrase is by depending on specific statistical models and examining the significance as it relates to the concept of a candidate term. The more relevant a candidate term is as a keyphrase, the more crucial it is to the analyzed document. Modern studies have broadened the scope of these techniques, and they are classified as supervised-based or unsupervised-based techniques [8].

The supervised key extraction method employs a model that has been trained on a set of key words. There are two groups of documents; one is used for training and the other for testing. They necessitate manual annotation in the learning dataset, which is time-consuming and inconsistent. Unfortunately, most authors only assign key words to their documents when they are required to. Supervised key extraction methods make use of training datasets, which are corpora of texts and their corresponding (i.e., previously assigned) keys. Some of the approaches are naive bayes [9], neural network-based [10] methods, while others investigate more sophisticated techniques, e.g., Genitor Extraction (GenEx) [4], and Keyphrase Extraction Algorithm (KEA) [11]. Even though KEA and GenEx performed similarly well, KEA, which was particular about extracting keyphrases from abstracts by employing naive bayes, has been shown to be more practical to implement and has served as the basis for other supervised keyphrase extraction methods.

Further work was done in [9] to improve on KEA [11] by the addition of linguistic knowledge to the representation (e.g., syntactic features). Instead of solely depending on statistics (e.g., term frequency and n-grams), quality keyphrases can be produced by adding linguistic knowledge. A better result was obtained as weighted by the previous assigned keyphrases by professional indexers. Specifically, high precision is achieved by extracting noun phrase chunks rather than n-grams. Also, an improvement was achieved irrespective of the term selection approach, by the addition of the part-of-speech tags allocated to the term as an attribute.

Also, a neural network-based [10], and Conditional Random Fields (CRF) [12] methods for extracting keyphrases out of scientific articles were also presented as comprehensive supervised-based approaches for extracting keyphrases automatically out of scientific articles. One of the flaws of the supervised key extraction method is that the different tag feature values for different parts of speech have no relationship. For instance, a singular noun has nothing in common with a plural noun and it does with an adjective.

Unsupervised methods for keyphrase extraction do not rely on labeled training data. Some of these methods are based on term statistics that assess the terms' degrees of importance to the document or collection of texts where they are found. Two commonly used term-weighting approaches in the literature are the word frequency and the Term Frequency and Inverse Document Frequency (TFIDF). Among several unsupervised-based key extraction approaches investigated, TFIDF was deemed the best among other statistical approaches [13], [14].

There are numerous approaches to unsupervised keyphrase extraction, the most basic of which relies entirely on frequency statistics, for instance, the Term Frequency and Inverse Document Frequency (TFIDF). Some of the approaches are: triplerank model [15], Keyphrase miner (Kpminer) [16], Keyword Extraction Using Collective Node Weight (KECNW) [17], and RankUp [18].

Further research on the unsupervised key extraction method, an innovative approach to key extraction using graph-based methods, was conducted [6], [7], [19]-[22]. Also, keyphrases were extracted from multiple perspectives (KIEMP) [23], the hyperbolic matching model [24], and the query-based model [25]. These approaches helped to improve the performance of extracting keyphrases. It was discovered that the proposed methods achieved better quality keyphrases than the state-of-the-art TFIDF method.

In summary, most keyphrase extraction models that have proven to be effective have employed a supervised-based approach, i.e., the models were trained on a document and tested in a particular domain. These systems are complex to use and mostly associated with human indexers. Most of these systems arouse attention and usually require specific protocols for their successful implementation and design. In the case of supervised learning, users have to worry about training the system with data sets corresponding to the domain and balancing the training set to be in tune with the documents to be tested. The consequence of this is that most users are not

patient enough in training the system with the data set. In the case of the unsupervised extraction approach, users need not train the system but depend on the system to produce a generic result. The consequence of this approach is that most unsupervised systems will select phrases by their default setting, i.e., the method installed by the system designer. The graph-based models that have proven to be successful did not consider whether a keyphrase contained a noun as part of it. The existing graph-based models that used the k-means algorithm had a challenge in setting the appropriate number of clusters as "k" is always unknown before extraction.

Therefore, an attempt was made in this study to use a different graph ranking algorithm to address the weak phrases that could be considered strong phrases and at the same time use a better clustering algorithm to solve the challenge of the k-means algorithm used.

## III. METHODOLOGY

This study developed a method and a generic system that automatically assigns keyphrases to documents and extracts generic keyphrases. The proposed method is an unsupervised approach that employs graph-modeling to extract keyphrases. It developed a method that creates a graph from a document, ranks it using a graph ranking algorithm, extracts phrases with nouns as part of them using a noun phrase identifier, and extracts similar phrases in clusters using a clustering algorithm. The weights in each stage were compared, and each candidate phrase was assigned a final weight.

### A. Data Collection and Preprocessing

The Hulth dataset, sourced from the Inspec database [9], was used in this study. This dataset contains abstracts of scientific papers from the inspec database, which has a total of 2000 abstracts. There are three files for each abstract: *.abstr*, *.contr* and *.uncontr*. A sample Hulth dataset is shown in Fig. 1. The file *.abstr* contains the title and the abstract. The file *.contr* contains the controlled, manually assigned keywords, separated by semicolons, while the file *.uncontr* contains the uncontrolled, manually assigned keywords, separated by semicolons.

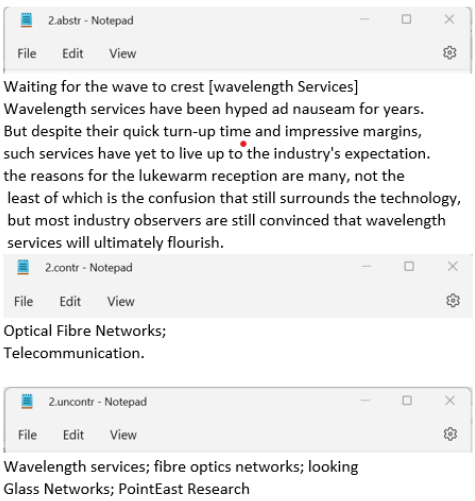


Figure 1. Sample dataset of Inspec database [9].

### B. Model Architecture

Fig. 2 depicts the proposed model architecture, which is an enhanced graph model for unsupervised keyphrase extraction in which the noun phrase identifier and affinity propagation algorithm are combined to extract quality keyphrases. In the preprocessing phase, the model accepts .txt, .pdf, and .doc file extensions as inputs. The document is then filtered using a list of stop words in the English language, such as "a, an, that." Punctuation marks like commas and full stops are removed. The stopwords that are removed are replaced with spaces in order to avoid incorrectly combining words. For example, "processed information is technology" cannot be combined as "processed in-formation" and "information technology," but rather "processed information" and "technology." The "/" symbol is used to replace full-stop signs and stop words in order to avoid combining the last word of one sentence with the first word of the next.

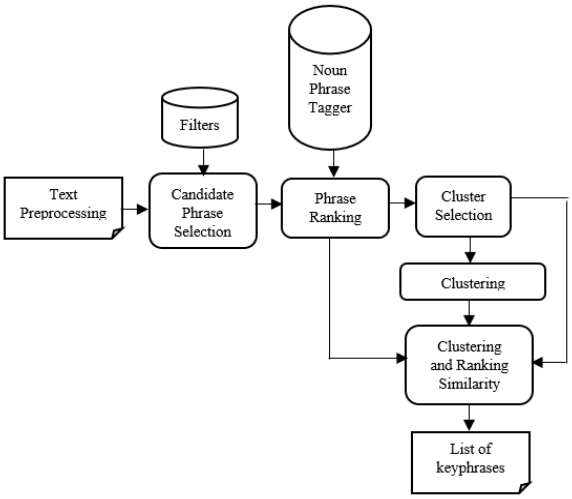


Figure 2. The graph-based model for noun-centric keyphrase.

Most authors assign keyphrases containing only two words in the literature. Therefore, in this model, the phrase combination is limited to two words within a sentence. After the words are combined, the phrases are built into the graph, where the phrases are nodes and the connections between phrases are relationships, i.e. phrases in the same sentence have a relationship, while phrases in different sentences do not. The textrank algorithm is used to rank the nodes; each node starts with the same value and either increases or decreases depending on the connection to it. Following the application of the textrank algorithm, each phrase is assigned a score.

The top n-ranks are stored for comparison with the clustering algorithms. The graph built from the textrank algorithm is parsed into the affinity propagation algorithm, which aids in clustering the text into groups. It sends the number of clusters to K-means in order to cluster the text, because affinity propagation chooses the number of clusters automatically, whereas K-means requires the user to choose the number of clusters. After the k-means algorithm has clustered the graph from the text rank algorithm, cosine similarity is used to determine which cluster in the k-means algorithm is closest to the n-rank

phrases from the text rank, and this cluster is chosen as a quality cluster in k-means. Cosine similarity is also used to determine the affinity propagation algorithm's closest cluster to the n-rank phrases from text rank, and this cluster is chosen as a quality cluster in affinity propagation. The intersections between the k-means cluster and the affinity propagation cluster are then selected as keyphrases. Fig. 3 depicts the block diagram for the proposed model.

Existing techniques used in formulating the proposed model include the textrank algorithm, noun phrase identifier, affinity propagation, k-means algorithm, and cosine similarity.

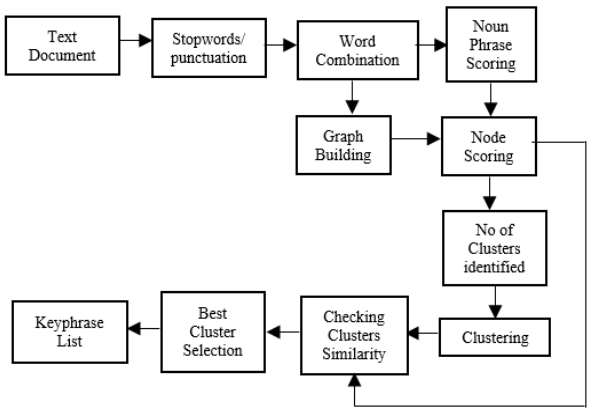


Figure 3. Block diagram for the graph-based model for noun-centric keyphrase extraction.

*1) Stopword list*

Stopwords are words that are filtered out before or after natural language data processing (text). This is a list of possible stopwords and punctuation marks in the English language. It covers a wide range of stopwords without becoming overaggressive or including an excessive number of words that a user might search for. There are 429 English words in this list. It contains articles such as "a," "an," and so on; pronouns like "he," "she," "they," and so on; prepositions like "under", "on," "below," and so on; conjunctions and interjections such as "but," "and," and so on. The stop words were extracted from (http://www.lextek.com/manuals/onix/stopwords1.html). The extracted stopwords were added to the Natural Language Toolkit (NLTK) stopwords list, resulting in a comprehensive list of stopwords.

*2) Noun phrase tagger*

The noun phrase tagger classifies a word or phrase as a noun or as containing a noun. The Oxford Advanced Learner's Dictionary, 7th Edition, was used to compile a list of nouns in English. The noun tagger used the noun list to determine whether a phrase contained a noun or was entirely a noun. Equation (1) depicts the formula for calculating the noun phrase score:

$$N_s = \frac{n_p}{w_p} \tag{1}$$

where $n_p$ is the number of nouns in a noun phrase, $w_p$ is the number of words in a noun phrase, and $N_s$ is the noun phrase score.

*3) Textrank algorithm*

Following the assignment of whether the words contain nouns or not, only the words that contain noun phrases are considered. Then a graph of the document is created, with each phrase acting as a node or vertices, *V*, and the relationships between the phrases acting as edges, *E*. The graph is then subjected to the algorithm. The algorithm traverses the list of nodes, collecting the influence of each of their inbound connections, and assigns each node a weight of one. The effect is typically only a connected vertex's value (usually 1, but tends to vary) and is then aggregated to establish a new score for the node, after which these scores are normalized, with the highest score becoming 1 and the remaining being scaled from 0 to 1 depending on their value. The process is repeated until the values stop changing as the algorithm gets nearer to the actual "value" for each node. Algorithm 1 depicts the textranking algorithm for keyphrase extraction.

**Algorithm 1: Pseudocode for Textrank Algorithm**

```
procedure TEXTRANK (D, k) //D is a set of documents and
        k is number of phrases to be extracted
    G ← BuildGraph(D) // Build network as explained
        above
    scores 4← (1.0,1. 0...1.0)// initialize scores
    converged ← False
    while converged == False do
        converged ←True
        oldScores ←scores
        for phrase c 1,2 ,, length(D):do
        //update phrase score according to rule given
            above
        scores[phrase] ← updatePhrase (G, phrase, d = 0
            scores)
        if | scores[phrase] – oldScore[phrase] | > e tl
            converged ←False
        end if
        end for
    end while
    return phrases with k highest scores
end procedure
```

This algorithm outputs key/important phrases for the cosine similarity based on the top n-scored phrases that have been considered critical. Given a directed graph *G = (V, E)* with a set of nodes (vertices) and edges and a vertex $v_i$, the score for $v_i$, $S(v_i)$, is computed iteratively until convergence. Equation (2) gives the mathematical formula as follows:

$$S(v_i) = (1-d) + d\left(\sum_{j\,\epsilon In(v_i)} \frac{S(v_j)}{Out(v_j)}\right) \tag{2}$$

where:

$In(v_i)$ represents a set of vertex v that are predecessors,

$Out(v_i)$ represents a set of vertex v that are successors, and *d* is a damping factor ranging from 0 to 1, and it was set to 0.85 [26].

*4) Affinity propagation algorithm*

The affinity propagation algorithm shown in Algorithm 2 receives the graph of phrases generated by the textrank algorithm. The algorithm is a clustering algorithm that sends messages between pairs of data points until a set of exemplars, each corresponding to a cluster, unfolds. The Affinity propagation algorithm accepts a real number, $s_{kk}$

as input for each data point *k*, which is termed a "preference." Data points with high $s_{kk}$ values are more likely to be exemplars.

**Algorithm 2: The Basic Affinity Propagation Algorithm**

given data point **i** and data point **k:**
**result1**= **α**
**for** each **data point z such that (z** ≠ k):
**temp1** = **avail [**i, **z] + simi [**i, z]
**if (temp1** > result):
**result1 =temp1**
resp[i, **k]= simi [i,k] – result1**
**if (i** ≠ k):
sum= 0
**for each data point z such that (z** ≠ **i) and (z** ≠k):
**temp2** = resp[z,k]
**if (tempt > 0**):
**sum = sum + tem**p2
**result2 = sum + resp[k,k]**
**if (result2** > 0):
result2 = 0
**else:**
sum = 0
**for** each **data point z such that (z** ≠ **k**):
result2 = resp[z, k]
**if (result2** > 0):
sum = sum + result2
result2 = sum
**final result = resultl + result2**
**end**

There are two types of messages that are conveyed between data points, each addressing a distinct type of competition. At any point during the process, messages can be aggregated to determine which points served as exemplars and which points belong to which exemplars. The "responsibility" $r_{ik}$ conveyed from data point *i* to candidate exemplar point *k* reflects the cumulative proof establishing the appropriateness of point *k* to serve as the exemplar for point *i* considering other prospective exemplars for point *i*. The "availability" $a_{ik}$, conveyed from candidate exemplar point *k* to point *i* represents the cumulative proof establishing the appropriateness of point *i* to choose point *k* as its exemplar, considering other points' claims that point *k* should be an exemplar, $r_{ik}$ and $a_{ik}$ can be regarded as log-probability ratios. The availabilities are set to zero at the start: $a_{ik} = 0$. Therefore, the responsibilities are calculated in Equation (3) as follows:

$$r_{ik} = s_{ik} - \max_{\substack{(i,k) \\ i \neq k}} (a_{ik} + s_{ik}) \tag{3}$$

where the "responsibility" updates the entire candidate exemplars to contend for the possession of a data point. The "availability" assembles indications from data points to determine whether a certain candidate exemplar will be a good exemplar. The formula for calculating a datapoint's availability is shown in Equation (4) as follows:

$$a_{ik} = min\{0, r_{kk}\} + \sum_{\substack{(i,k) \\ i \neq k}} max\{0, r_{ik}\} \tag{4}$$

The availability $a_{ik}$ is equal to the total of the positive responsibilities a candidate exemplar *k* received from other points plus the self-responsibility $r_{kk}$. Because a good exemplar only needs to describe some data points well (positive responsibilities), irrespective of how badly it describes other data points (negative responsibilities), only the positive portions of incoming responsibilities are introduced. If point *k's* self-responsibility $r_{kk}$ is negative (denoting that point *k* is presently best placed as a member of another exemplar than as an exemplar), point *k's* availability as an exemplar can be enhanced if some other points have positive responsibilities for point *k* as their exemplar. To minimize the effect of substantial inbound positive responsibilities, a threshold is assigned to the total sum so as not to exceed zero. The "self-availability" $a_{kk}$ is updated in a different way as shown in Equation (5) as follows:

$$a_{kk} = \sum_{\substack{(i,k) \\ i \neq k}} max\{0, r_{ik}\} \tag{5}$$

Given the positive responsibilities sent to candidate exemplar *k* from other points, this message demonstrates cumulative proof that point *k* is an exemplar. The preference values and the message-passing procedure have an effect on the number of clusters. The n-rank phrases from the textrank algorithm were compared with each cluster in the affinity propagation. The cluster that contains most of the n-rank phrases is selected as the best cluster by affinity propagation. The criterion matrix for *i* and *k* is given in Equation (6) as follows:

$$c_{ik} = r_{ik} + a_{ik} \tag{6}$$

*5) K-means algorithm*

The k-means clustering method is a technique for describing the "best" partitioning of a dataset with k clusters. Its objective function is to minimize the sum of all squared distances within a cluster, for all clusters. The K-means algorithm shown in Algorithm 3 is also referred to as the Lloyd's algorithm. The objective function is defined in Equation (7) as follows:

$$\arg \min_{i} E(C) = \sum_{i=1}^{N} ||x_i - c_i\ ||^2 \tag{7}$$

where, *C* represents a cluster ($c_1$, $c_2$, ...$c_n$) and;
$c_i$ is the centroid closest to the sample data point $x_i$.

**Algorithm 3: K-means Algorithm**

**Input:**
**D = {t₁ t₂ …tₙ** // Set of elements
K = //number of desired clusters
**Output**:
**K // set of cl**usters
**K-means algorithm**
**Arbitrari**ly choose k objects from D a**s the initial clusters cent**
**Assign initial values for m₁, m₂, … mₖ**
**Repeat:**
1. **(re)assign each object to the cluster to which the object most similar, based on the mean value of the objects in clusters:**
2. **2. Update the cluster means. i.e. calculate the mean valu of the objects for each cluster**

**Until no change in clusters**

The number of clusters in k-means is fixed when clustering occurs, which distinguishes it from many other clustering methods. This can be viewed as both a strength and a weakness. The k-means method does not introduce new clusters in the case of an anomaly data point because it has a fixed number of clusters; instead, it categorizes the anomaly data point into its closest cluster. The disadvantage of using a fixed number of clusters is that it

is sometimes difficult to tell how many clusters a dataset contains. Using an inappropriate *k* may cause the k-means method to produce poor results, possibly rendering it useless.

*6) Cosine similarity*

The cosine similarity is a similarity measure that checks the degree of similarity of two entities. The entities could be words, sentences, documents, etc. The cosine similarity used in this work calculates the similarity between the n-ranked phrases from the text rank algorithm with each clustering algorithm (Affinity propagation and the k-means algorithm). The n-ranked phrases are assumed to be a cluster and are compared with each cluster of the two algorithms. The most similar clusters in affinity propagation and k-means algorithms are compared with each other to identify the similarity. The intersection of the k-means algorithm and affinity propagation clusters are deemed keyphrases. The mathematical formula is shown in Equation (8) as follows:

$$cosine\ similarity = \frac{x_i x_j}{\|x_i\|\|x_j\|} \quad (8)$$

where $x_i$ and $x_j$ are the two entities. The cosine similarity takes the product of the two entities divided by the product of their magnitudes.

## C. Model Formulation

In this study, there are two approaches to the keyphrase extraction model, namely, the ranking approach and the clustering approach. The ranking approach is formulated using the textrank algorithm and noun phrase identifiers, while the clustering approach is formulated using the k-means algorithm and affinity propagation. The cosine similarity is only used to compare the ranked phrases and the clustered phrases. Algorithms 4 show the detailed the formulation approaches are described as follows:

*1) Ranking approach*

The ranking approach uses the textrank algorithm and noun phrase identifier in formulating the ranking model. After the words have been combined to a maximum of two words, the graph of the phrases' connections is plotted. The score of an individual phrase is calculated using the noun phrase identifier given in Equation (1) and the textrank algorithm in Equation (2). The new score of each phrase is computed in Equation (9) as follows:

$$Phrase\ Rank\ Score = S(p) + N_p \quad (9)$$

where, *S(p)* is the textrank phrase score after convergence, and;

$N_p$ is the noun phrase score.

The phrase rank score is the score of each phrase after computation. Each phrase is ranked in descending order of phrase rank score. The top n-rank phrases are assumed to be a cluster $C_{tn}$.

*2) Clustering approach*

The clustered phrases from the ranking approach are stored and compared with the two clustering algorithms. The affinity propagation algorithm takes the graph plotted by textrank and clusters it based on its algorithm. The resulting clusters are sent to k-means to enhance their clustering. From Equation (5), the number of clusters is equal to the number of exemplars. The number of clusters from the affinity propagation is represented by nCa which is the number of clusters gotten from affinity propagation. The cluster groups in affinity propagation are defined in Equation (10) as follows:

$$C_a \leftarrow c_1, c_2, \ldots, c_n \quad (10)$$

**Algorithm 4: Pseudocode for the Developed Algorithm**

**Input:** Document D = {$w_1, w_2$, ..., $w_n$}, S = {stopwords_list}, Q = {punctuation), {Nounlist}
Replace S and Q from D with "/"
Combine w in D until "/"
D = {$p_1$, $p_2$, ..., $p_n$}
  **If** N in D **then**
  NS = no of nouns in phrase / no of words in phrase
  **end if**
G ← BuildGraph(D)
Scores ← (1.0, 1.0, 1.0) // initialize scores
converged← false
  **while** converged == false **do**
    converged ←true
    old scores ←scores
      **for** phrase p ε 1, 2..., length(D): **do**
      //update phrase score according to rule given abov
      Scores [Phrases← UpdatePhrase (G, Phrase; d=0.
scores) if
        **If** | scores[phrase] - oldscores[phrase] | > ε **t**
        converged ←false
        **end if**
      **end for**
  **end while**
return TS // phrases with textrank scores
TN = TS + NS // textrank scores and noun phrase scores
given two datapoints i and k // these are the TS scores of differer
phrases result1 = α
  **for** each data point z such that (z≠k): // z is actually not i o
    temp1 = avail[i,z] + simi[i,z]
    **if** (temp1 > result);
    result1 = temp1
    resp[i,k] = simi[i,k] - result1 // responsibility matrix
**if** i≠k:
  sum = 0
  **for** each datapoint z such that (z≠i) and (z≠k):
    temp2 = resp[k,k]
    **if** (temp2 > 0):
      sum = sum + temp2
      result2 = sum + resp[k,k]
      **if** (result2 > 0):
        result2 = 0
**else**:
  sum=0
  **for** each data point z such that (z≠k):
    result2 = resp[z,k]
    **if** (result2 > 0):
      sum = sum + result2
      avail(i,k) = sum // availability matrix
      crite(i,k) = v resp(i,k) + avail(,k) */criterion matri
objects having similar results in their row are in th
same exemplars*/
AP = ($AP_1$, $AP_2$...,,$AP_s$) // no of exemplars or clusters
Arbitrarily choose AP objects from TS as the initial clusters cent
Assign initial values for $AP_1$, $AP_2$, . $AP_s$;
**Repeat**:
1. (re)assign each object to the cluster to which the object is n similar, based on the mean value of the objects in the cluste
2. Update the cluster means, i.e., calculate the mean value of objects for each cluster

**Until:**
no change in clusters KM = $KM_1$.$KM_2$…, $KM_z$ //v Kmeans clust
Maximum cosine similarity1 = (TN • AP) / ( | TN | * | AP | )
Maximum cosine similarity2 = (TN • AP) / ( | TN | * | AP | )
Phrase p = maximum cosine similarity1 Intersection maximum cosine similarity2

## IV. Results and Discussions

The developed model was simulated and evaluated in the Anaconda Python version 3.6 environment. The codes for the implementation of the keyphrase extraction model were written using the Scientific Python Development Environment (Spyder), an Integrated Development Environment (IDE) included with Anaconda. Other Python modules used for analysing the model are Tika, Tkinter, NumPy, Math, NLTK, Re, String, Itertools and K-means. The dataset [9] used in this study, contained 500 documents of author-assigned keyphrases from the Inspec database. The dataset was split into three partitions in which 1000 abstracts were used for training, 500 for validation, and the remaining 500 were used for testing. The k-fold cross-validation method was used to validate the dataset to avoid overfitting. The study employed an unsupervised approach, i.e., the study used the test set to establish a detailed comparison with the selected existing work [6]. In the dataset, phrases that were not in the abstract but were deemed appropriate by a human expert are stored in *.uncontr* as keyphrases. An output keyphrase was considered a valid keyphrase if it was similar to a manually assigned keyphrase in *.uncontr*.

Candidate phrases were selected, and then phrase scoring was performed using some Python modules. The phrase scoring involves scoring a selected candidate phrase using the textrank algorithm, and if the phrase contains a noun, part of it. The noun phrase score and textrank score were calculated differently and later combined to give the phrase a unique score. The number clusters were detected by implementing Affinity Propagation on the scores of the phrases from the textrank implementation. The cluster groups identified were further clustered using the Affinity Propagation and k-means algorithms. The clustered groups were used as the k in the k-means algorithm implementation. The detailed results are as follows:

### A. Simulation Results

The simulation result of the keyphrase extraction process is shown in Table I. The document column contains the individual documents; the author-assigned column contains the number of author-assigned keyphrases; the keyphrase extractor is the developed model extraction; the true positive is the number of phrases agreed upon by the developed model and the authors as keyphrases; the false positive is the number of keyphrases the developed model was unable to identify; and the false negative is the number of keyphrases the model extracted but were not identified as keyphrases by the author. The true positive, false positive, and false negative were used to calculate the precision and recall. The precision, which is a measure of the model's exactness when compared to the author-assigned, which was the ground truth, was calculated by dividing the keyphrases that matched the author-assigned keyphrases by the number of author-assigned keyphrases. The recall, which is a measure of the model's correctness based on model extraction, was calculated by dividing the total number of keyphrases extracted that match the author-assigned keyphrases by the total number of keyphrases extracted by the model. The f-measure is calculated as a weighted average of precision and recall.

The first twenty documents are displayed out of a possible five hundred, and the author-assigned keys are not fixed because they contain between four and nineteen keyphrases. The developed model extracted ten keyphrases per document.

TABLE I. Simulation Results of the Keyphrase Extraction Model

| D | AA | KE | TP | FP | FN | Precision (%) | Recall (%) | F-measure (%) |
|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 10 | 2 | 2 | 8 | 50 | 20 | 28.57 |
| 2 | 4 | 10 | 4 | 0 | 6 | 100 | 40 | 57.14 |
| 3 | 4 | 10 | 2 | 2 | 8 | 50 | 20 | 28.57 |
| 4 | 10 | 10 | 5 | 5 | 5 | 50 | 50 | 50 |
| 5 | 7 | 10 | 4 | 3 | 6 | 57.14 | 40 | 47.06 |
| 6 | 9 | 10 | 5 | 4 | 5 | 55.56 | 50 | 52.63 |
| 7 | 19 | 10 | 6 | 13 | 4 | 31.58 | 60 | 41.38 |
| 8 | 8 | 10 | 6 | 2 | 4 | 75 | 60 | 66.67 |
| 9 | 19 | 10 | 4 | 15 | 6 | 21.05 | 40 | 27.58 |
| 10 | 6 | 10 | 5 | 1 | 5 | 83.33 | 50 | 62.5 |
| 11 | 7 | 10 | 3 | 4 | 7 | 42.86 | 30 | 35.29 |
| 12 | 8 | 10 | 5 | 3 | 5 | 62.5 | 50 | 55.56 |
| 13 | 16 | 10 | 2 | 14 | 8 | 12.5 | 20 | 15.38 |
| 14 | 8 | 10 | 4 | 4 | 6 | 50 | 40 | 44.44 |
| 15 | 14 | 10 | 4 | 10 | 6 | 28.57 | 40 | 33.33 |
| 16 | 8 | 10 | 4 | 4 | 6 | 50 | 40 | 44.44 |
| 17 | 12 | 10 | 5 | 7 | 5 | 41.67 | 50 | 45.45 |
| 18 | 7 | 10 | 4 | 3 | 6 | 57.14 | 40 | 47.06 |
| 19 | 10 | 10 | 5 | 5 | 5 | 50 | 50 | 50 |
| 20 | 6 | 10 | 1 | 5 | 9 | 16.67 | 10 | 12.5 |

LEGEND: D= Document, AA= Author-assigned, KE= Keyphrase Extractor, TP= True Positive, FP=False Positive, FN=False Negative.

Its true positive score ranges between one and six across the twenty documents, its false positive score ranges between zero and fifteen, and its false negative score ranges between four and nine.

Table II displays the average result from the 500 documents. The author-assigned keyphrases across the 500 documents were an average of 9.44 keyphrases. The developed model extracted an average of 10 keyphrases. The average true positive across the 500 documents was 4.58, the average false positive was 4.86, the average false negative was 5.42, the average precision was 48.5 %, the average recall was 45.8%, and the average F-measure score was 47.1%. These results demonstrated that the model was capable of identifying keyphrases that are similar to the author-assigned keyphrases.

TABLE II. Weighted Average Results of the Developed Model Result across the 500 Documents

| D (500) | AA | KE | TP | FP | FN | Precision (%) | Recall (%) | F-measure (%) |
|---|---|---|---|---|---|---|---|---|
| 500 docs | 9.44 | 10 | 4.58 | 4.86 | 5.42 | 48.5 | 45.8 | 47.1 |

LEGEND: D= Document, AA= Author-assigned, KE= Keyphrase Extractor, TP= True Positive, FP=False Positive, FN=False Negative.

### B. Evaluation Results

To determine the effectiveness of the developed model using the dataset, the model's performance was evaluated by comparing the number of keyphrases matching the author-assigned keystrokes generated by the developed

model to the performance of a selected existing model [6], with precision, recall, and f-measure as performance metrics. The extraction sample for three documents is shown in Table III. The table displays the number of keyphrases that match the author-assigned keyphrases.

TABLE III. EXTRACTED KEYPHRASES FROM AUTHOR-ASSIGNED THREE DIFFERENT DOCUMENTS

| | |
|---|---|
| DOCUMENT ONE | |
| Document's Title | Waiting for the wave to crest [wavelength services] |
| Existing model | wavelength services; wavelength; technology; crest wavelength; services, crest |
| Developed model | wavelength services; technology, confusion, reasons, industry observers, services, years, hyped ad nauseam, lukewarm reception |
| | |
| DOCUMENT TWO | |
| Document's Title | NuVox shows staying power with new cash, new market |
| Existing model | Telecom market; new market; NuVox Communications; credit facility new, market, NuVox |
| Developed model | NuVox communication; today; new market; raise cash; NuVox, telecom market; new credit facility; million, says; long run |
| | |
| DOCUMENT THREE | |
| Document's Title | Analyzing the benefits of 300mm conveyor-based AMHS |
| Existing model | Initial conditions; switching line; output samples; design; output feedback; fast output sampling feedback; system state; output feedback sliding; switching line; systems; designed; design; designing; initial; initially |
| Developed model | Initial conditions; simulation results; discrete output feedback; idea; system state; require; paper; output samples; work design |

As shown in Table III, the first document shows that the developed and existing models generated two and three keyphrases (i.e. the underlined) that corresponded to the author-assigned keyphrases, respectively. The second document demonstrated that the developed model generated four keyphrases, whereas the existing model generated only two keyphrases. The third document shows that the developed model and the existing model generated five and three keyphrases that correspond to the author-assigned keyphrases, respectively.

For instance, in the document one, the true positive for the existing model is two (the author and model agree on "wavelength services" and "crest" as keyphrases), and the true positive for the developed model is also two (the author and model agree on "wavelength services" and "technology" as keyphrases).

The existing model has two false positives (the author agrees on "wavelength services" and "technology" as keyphrases but the model disagrees), and the developed model has two false positives (the author agrees on "wavelength" and "crest" as keyphrases but the model disagrees). The false negative for the existing model is one (the model agrees on "services" as a keyphrase and the author disagrees), and the false negative for the developed model is eight (the model agrees on "confusion", "reasons", "industry observers", "services", "years", "hyped ad nauseam" and "lukewarm reception" as keyphrases and the author disagrees). Thus, the evaluation result for document one showed that the developed model had a precision of 75% and a recall of 66.7%, whereas the existing model had a precision of 50% and a recall of 20%.

TABLE IV. AVERAGE PRECISION, RECALL AND F-MEASURE SCORES

| | Precision (%) | Recall (%) | F-measure (%) |
|---|---|---|---|
| Existing model | 43.0 | 40.2 | 41.6 |
| Developed Model | 48.5 | 45.8 | 47.1 |

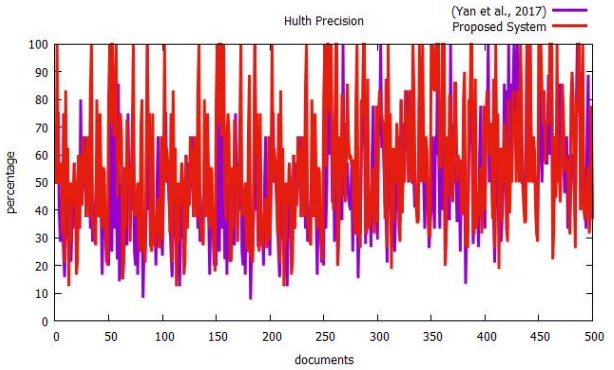


Figure 4. Precision evaluation result.

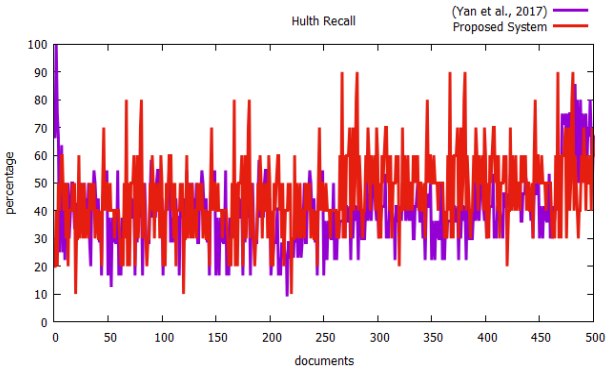


Figure 5. Recall evaluation result.

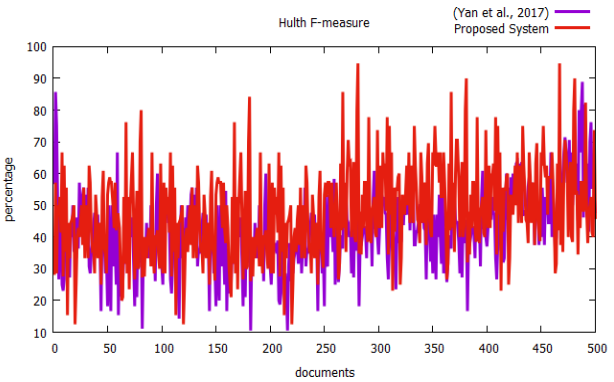


Figure 6. F-measure evaluation result.

Table IV shows the average precision, recall, and f-measure of the developed model against the existing model over 500 documents. Fig. 4, Fig. 5, and Fig. 6 show the precision, recall, and f-measure evaluation graph results for the two models, which are further broken down in Fig. 7, Fig. 8, and Fig. 9 for easy assessment. The evaluation results show that the developed model outperforms the

existing model by 5.5% in precision, 5.6% in recall, and 5.5% in f-measure. The developed model clearly outperformed the existing model across 500 documents for the majority of the iterations done. This significant performance of the developed model is due to the inclusion of affinity propagation, which selects the number of clusters, and the noun phrase identifier, which determines whether a phrase contains a noun.

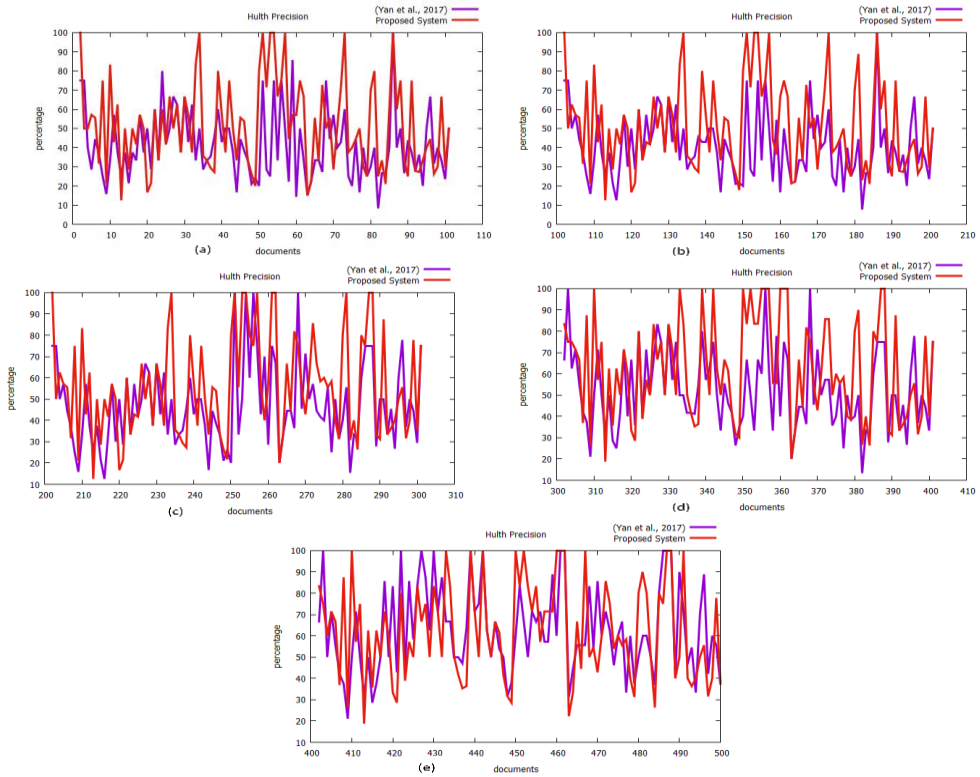


Figure 7. Precision evaluation results (a) 1-100 documents, (b) 101-200 documents, (c) 201-300 documents, (d) 301-400 documents, (e) 401-500 documents.

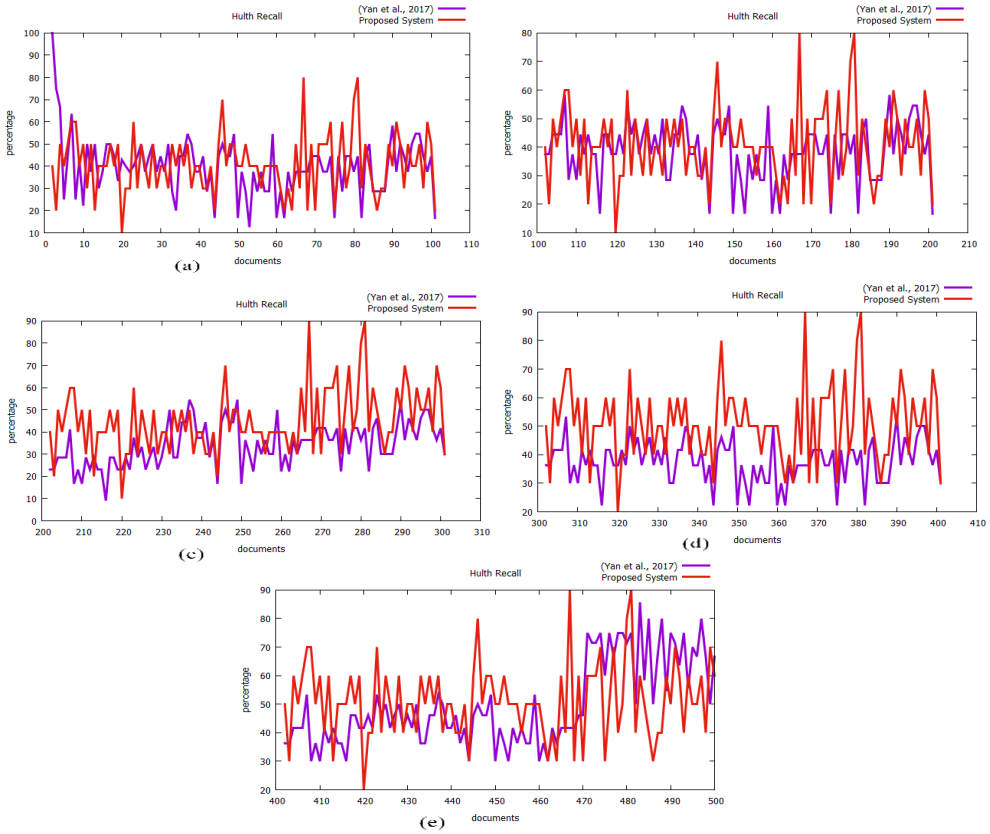


Figure 8. Recall evaluation results (a) 1-100 documents, (b) 101-200 documents, (c) 201-300 documents, (d) 301-400 documents, (e) 401-500 documents.

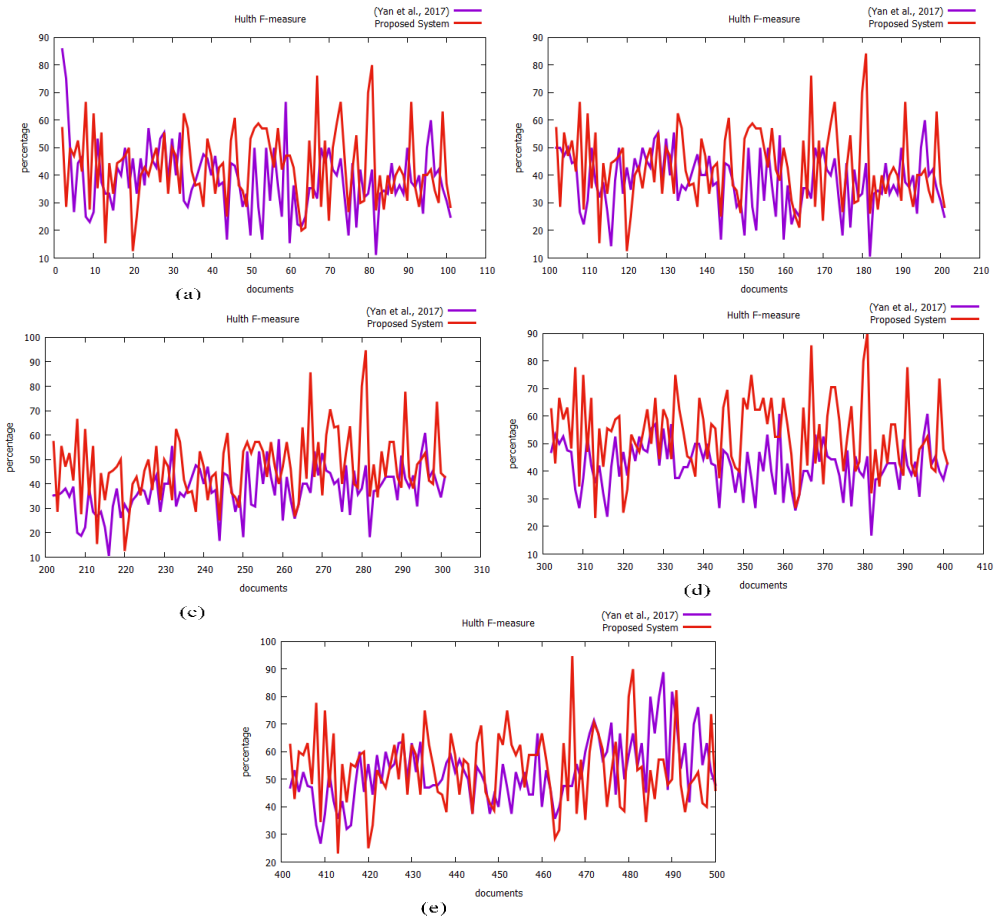


Figure 9. F-Measure evaluation results (a) 1-100 documents, (b) 101-200 documents, (c) 201-300 documents, (d) 301-400 documents, (e) 401-500 documents.

## V. Conclusion

This research developed and tested a noun-centric keyphrase extraction method for English language documents. The study concluded that the use of a noun phrase identifier, textrank algorithm, affinity propagation, and k-means algorithm resulted in good precision, recall, and f-measure performance. The textrank algorithm was critical to the extraction process as it modelled the document as a graph. The noun phrase identifier is very important in extracting keyphrases because most keyphrases contain nouns.

In this study, the method employed is scalable, as new nouns could be added to the list to improve on the existing one. The model demonstrated an innovative and user-friendly method of extracting keyphrases without complication. It operates in such a way that it can extract keyphrases from documents regardless of their domain. Finally, this study has introduced an invaluable method for academic researchers in ensuring reliable keyphrase extraction in journal articles, which would translate into efficient keyphrase extraction.

The model developed in this study would be extremely useful to the academic community. It is suggested for use on academic journal websites and by academic researchers to extract keyphrases from journal articles. It can also be used to generate headlines for documents or newspapers, to summarize text in a document, to improve search precision, and to generate the back of a book index. In future research, phrases containing numbers should not be ignored because some author-assigned keyphrases may contain numbers that were removed during the preprocessing phase of the developed model. Also, domain knowledge could be incorporated to create a domain-specific model that employs a graph-ranking algorithm and a noun-identifier. Furthermore, while the focus of this study was on nouns, other parts of speech can be incorporated to produce higher-quality keyphrases.

## Conflict of Interest

The authors declare no conflict of interest.

## Author Contributions

Under the supervision of I. O. Awoyelu and F. O. Hunsu, R. O. Abimbola formulated the research goals and objectives, designed the methodology, and created and evaluated the models. R. O. Abimbola and B. O. Akinyemi were responsible for the preparation of the published work, specifically writing the initial draft, critical review, and writing revision. G. A. Aderounmu was in charge of the study's oversight and leadership responsibility for the research activity planning and execution, as well as mentorship of the core team.

## ACKNOWLEDGMENT

This Research was funded by the TETFund Research Fund" and Africa Centre of Excellence OAK-Park.

## REFERENCES

[1] K. M. Hammouda, D. N. Matute, and M. S. Kamel, "Corephrase: keyphrase extraction for document clustering," in *Proc. the International Workshop on Machine Learning and Data Mining in Pattern Recognition*, 2005, pp. 265-274.
[2] E. Youn and M. K. Jeong, "Class dependent feature scaling method using naive bayes classifier for text datamining," *Pattern Recognit. Lett.*, vol. 30, no. 5, pp. 477-485, 2009.
[3] X. Wan, J. Yang, and J. Xiao, "Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction," in *Proc. the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic, 2007, pp. 552-559.
[4] P. D. Turney, "Coherent keyphrase extraction via web mining," in *Proc. the Eighteenth Research Council*, 1999, pp. 23-32.
[5] I. S. Gerardo, M. Cristian, V. M. Maria, *et al.*, *Ontology-Based Data Access Leveraging Subjective Reports*, 1st ed., Springer International Publishing, 2017.
[6] Y. Yan, Q. Tan, Q. Xie, *et al.*, "A graph-based approach of automatic keyphrase extraction," *Procedia Computer Science*, vol. 107, no. 1, pp. 248-255, 2017.
[7] I. O. Awoyelu, R. O. Abimbola, A. T. Olaniran, *et al.*, "Performance evaluation of an improved model for keyphrase extraction in documents," *Computer Science and Information Technology*, vol. 4, no. 1, pp. 33-43, 2016.
[8] K. S. Hasan and V. Ng, "Automatic keyphrase extraction: A survey of the state of the art," in *Proc. the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014, pp. 1262-1273.
[9] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proc. the Conference on Empirical Methods in Natural Language Processing*, 2003, pp. 216-223.
[10] K. Sarkar, M. Nasipuri, and S. Ghose, "A new approach to keyphrase extraction using neural networks," *International Journal of Computer Science Issues*, vol. 7, no. 1, pp. 16-25, 2010.
[11] E. Frank, W. P. Gordon, H. W. Ian, *et al.*, "Domain-Specific keyphrase extraction," *Machine Learning*, pp. 668-671, 1999.
[12] P. Bhaskar, K. Nongmeikapam, and S. Bandyopadhyay, "Keyphrase extraction in scientific articles: A supervised approach," in *Proc. Computer Linguistic (COLING) 2012 Demonstration papers*, 2012, pp. 17-24.
[13] L. Teixeira, G. Lopes, and R. A. Ribeiro, "Automatic extraction of document topics," *International Federation for Information Processing*, pp. 101-108, 2011.
[14] F. Liu, F. Liu, and Y. Liu, "A supervised framework for keyword extraction from meeting transcripts," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 3, pp. 538-548, 2011.
[15] T. Li, L. Hu, H. Li, *et al.*, "TripleRank: An unsupervised keyphrase extraction algorithm," *Knowledge-Based Systems*, vol. 219, pp. 1-12, 2021.
[16] S. R. El-Beltagy and A. Rafea, "KP-Miner: A keyphrase extraction system for English and Arabic documents," *Information Systems*, vol. 34, no. 1, pp. 132-144, 2009.
[17] S. K. Biswas, M. Bordoloi, and J. Shreya, "A graph based keyword extraction model using collective node weight," *Elsevier Expert Systems with Applications*, vol. 97, no. 1, pp. 51-60, 2018.
[18] G. Figueroa, P. Chena, and Y. Chena, "RankUp: Enhancing graph-based keyphrase extraction methods with error-feedback propagation," *Computer Speech & Language*, vol. 47, no. 1, pp. 112-131, 2017.
[19] C. Mallick, A. K. Das, M. Dutta, *et al.*, "Graph-Based text summarization using modified TextRank," *Soft Computing in Data Analytics*, vol. 10, no. 1, pp. 137-147, 2019.
[20] W. Shi, W. Zheng, J. X. Yu, *et al.*, "Keyphrase extraction using knowledge graphs," *Data Science and Engineering*, vol. 2, no. 4, 2017.
[21] H. Yeom, Y. Ko, and J. Seo, "Unsupervised-Learning-Based keyphrase extraction from a single document by the effective combination of the graph-based model and the modified C-value method," *Computer Speech & Language*, vol. 58, no. 1, pp. 304-318, 2019.
[22] M. S. El-Bazzi, D. Mammass, T. Zaki, *et al.*, "A graph-based ranking model for automatic keyphrases extraction from Arabic documents," in *Proc. Industrial Conference on Data Mining*, 2017, pp. 313-322.
[23] M. Song, L. Jing, and L. Xiao, "Importance estimation from multiple perspectives for keyphrase extraction," arXiv preprint arXiv:2110.09749, 2021.
[24] M. Song, Y. Feng, and L. Jing, "Hyperbolic relevance matching for neural keyphrase extraction," arXiv preprint arXiv:2205.02047, 2022.
[25] M. Docekal and P. Smrz, "Query-Based keyphrase extraction from long documents," arXiv preprint arXiv:2205.05391, 2022.
[26] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, pp. 107-117, 1998.



**Rilwan O. Abimbola** holds a B.Sc. (2010) in Computer Science from Babcock University Ilishan-Remo, an M.Sc. (2016) and a Ph.D. (2021) in Computer Science from Obafemi Awolowo University Ile-Ife. He is a member of the Nigeria Computer Society. He is a Lecturer in the Department of Computer Sciences at Technical University, Ibadan, and a member of the Data Communication Group in the Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife. His major research areas are in human language processing, information retrieval, data mining, and application development. He has published articles in reputable journals and has presented papers at learned conferences.

**Iyabo O. Awoyelu** is an Associate Professor of Computer Science in the Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria. She teaches Database Systems, Principles and Applications of Data Mining, Techniques in Data Analysis, and Principles of Compilers. She has over 20 years of teaching and research experience. Her research interests are in data warehousing and data mining. She is a member of the Nigerian Computer Society (NCS), Nigerian Women in Information Technology (NIWIT), and Computer Professionals of Nigeria (CPN). She was on research visitation to the Indian Institute of Technology (IIT), Hyderarbad, India in November and December, 2019. She was the Vice Dean of the Faculty of Technology at OAU from July 2020 to June 2021. She has mentored a number of postgraduate students and has over thirty (30) published journal articles and referred conference proceedings to her credit.

**Folasade O. Hunsu** is a senior lecturer in the Department of English at Obafemi Awolowo University, Ile-Ife, Nigeria. She teaches the use of English and other courses in literature. Her research interests are in Women's Studies, Feminist Activism, Autobiography, and African Literature. She has won prestigious academic fellowship awards such as the British Academy Fellowship, Study of United States Institutes, and African Humanities Program. Her essays have appeared in reputable journals, including JENdA: A Journal of Culture and African Women's Studies and a/b: Auto/Biography Studies. Hunsu is working on a monograph on African female academics and feminism.

**Bodunde O. Akinyemi** holds B.Tech. (2005) in Computer Science from Ladoke Akintola University Ogbomosho, M.Sc. (2011) and Ph.D. (2014) in Computer Science from Obafemi Awolowo University Ile-Ife. She is a member of the International Association Engineers, IEEE, Nigerian Computer Society (NCS), Nigerian Women in Information Technology (NIWIT), and Computer Professionals of Nigeria (CPN). She is a Senior Lecturer and member of the Data Communication Group in the Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife. Her major research areas are in Data Communication, Network Security and Performance management, Software Development and Blockchain Technology. She has published more than 30 articles in reputable journals and learned conferences.

**Ganiyu A. Aderounmu** is a professor of Computer Science and Engineering from Obafemi Awolowo University, Ile-Ife, Nigeria. He is a Full member of the Nigeria Society of Engineers (NSE) and also a registered Computer Engineer with Council for Regulation of Engineering Practice in Nigeria (COREN). He is also a Full member of Nigeria Computer Society (NCS) and Computer Professional Registration Council of Nigeria (CPN). He has over 30years of experience in teaching and research. He is an author of many journal articles in Nigeria and abroad. His special interest includes computer communication and network. He is a visiting Research Fellow to the University of Zululand, Republic of South Africa. He was the former head of the Department of computer Science & Engineering, former Dean, Faculty of Technology, former President of Nigeria Computer Society (NCS), and the former Director of Information Technology and Communication Unit (INTECU).