Hashing And Clustering Based Novelty Detection

[1]Pooja Goyal, [2]Sushil Kumar, [3]Komal Kumar Bhatia

[1]Research scholar, [2]Assistant Professor, [3]Professor
[123]Department of Computer Engineering,
[123]J.C Bose University of Science & Technology YMCA, Faridabad, India

Abstract: *Novelty detection is a method of identifying new data from the incoming stream of documents. Many technologies for novelty detection are available at present. To enhance the performance of novelty detection, we present another method for novelty detection. In this work, we organize a new structure for novelty detection using hashing and clustering. This method consists of two stages. The first phase consists of dividing sentences, stopwords removal, text preprocessing, converting the N-grams into hashes from adaptations of documents. The second phase consists of clustering and novelty detection based on a fixed threshold. This method is useful for determining the high number of novel documents by detecting new information in data. This method is useful for finding excess in documents and also providing relevant documents against the query. The main aim of this work is to find appropriate and redundant free documents. The goal is to provide information to the user as fast as feasible.*

IndexTerms: *Clustering, Hashing, Novelty Detection, Plagiarism.*

I. INTRODUCTION

A substantial number of documents are obtainable on the internet with the improvement of the web. To deal with a massive amount of data, Novelty Detection has become an enormous part. To deal with and coordinate a vast number of documents, Novelty Detection is a key technology. Documents or sentences with fresh content can be referred to as novel information. The process of attaining novel information is Novelty Detection. TheProcess of Novelty Detection is as follows:

1. Preprocessing: This principally involves the dismissal of stopwords and performing stemming.
2. Categorization: In this phase documents or sentences are classified into the relevant bin.
3. Novelty Detection: The bin is explored by Novelty Detection module for novel information.

Novelty Detection has its numerous applications such as Fraud Detection, Insurance, Railways, and Robotics etc. The web presents us with a plentiful of information but it has negative attributes too. The negative attribute is that it contains a lot of repetitive data. People may notice copied data over and over. This is because copying data is the effortless way to complete the task. Plagiarism can be said as redundant data. Redundancy makes challenging for evaluation of novel documents. Many routines have been constituted for novelty detection. The major issue has been the precise measurement of relevant and redundancy-free documents due to practices used for similarity and novelty computations. This algorithm is based on character similarity. In this algorithm, the similarity check is conducted on characters. Two documents are analyzed similar if they have similar content in terms of strings. Novelty between two documents is determined as the minimum similarity between two documents utilizing some fixed threshold. This work focus on the character-based algorithm rather than a sentence based algorithm for document matching.
The principles of this work are as follows:

1. The documents to be used are presented with an extension .doc, .docx and/or .txt.
2. The novelty detection process commences with document pre-processing.
3. Pre-processing mainly comprises tokenization, space removal, stemming, stopwords removal.
4. N-grams are generated from tokens in order to obtain documents with fixed length strings.
5. N-grams are further processed for discovering hashes. Hashes are collected in order to diminish the size of documents.
6. The strings are reformed to some numeric values called hashes.
7. A suitable similarity measure is applied to hashes for similarity determination.
8. Clustering is performed in order to group similar documents together and non-similar documents together. Various Clusters are formed. Each cluster contains documents which are different from documents of other clusters but are similar to documents within a cluster.
9. On the basis of some measures, a Cluster Head is selected. Cluster Head will represent the whole cluster.

10. In the end, Novelty is concluded using a similarity coefficient.

In this work, we paid consideration to two points: first, with the help of similarity measure computation on hashes, clustering is done; second, determining novel documents with the help of cluster head picked from clusters. The main aim is to locate relevant and redundancy-free documents having novel information.

II. RELATED WORK

A. Topic-based novelty detection

The work done in Topic-based novelty detection is entirely explained by S.Sendhilkumar, Nachiyar N Nandhini [1]. Preprocessing is performed on input section using latent Dirchilet algorithm and Hierarchical Pachinko Allocation Model the preprocessed input is topic modeled. Novelty score is greatly influenced by topic modeling. This is because in terms of topic, subtopics which are classified by LDA and home, the novelty is considered. The corpus was compared with the topics which are retrieved from topic modeling. The documents are retrieved which have topics, subtopics which were identified by topic modeling. Their semantic is also retrieved. LDA and ham are used for the retrieval process. Clustering is achieved on documents using term frequency. The cluster contained the documents which had similar TF/IDF. The number of topics formulates the number of clusters to be formed. To find the most relevant documents, clustering is done. It also helps in reducing size. Similarity check is performed on documents within a corpus. Cosine similarity is used for computing similarity. The outline of input data is obtained and is idea mapped with documents which were treated by mapping. Concept mapping shows the correlation between documents which are served as concepts. The concept may represent a word, idea or a topic. The linking method between concepts is represented by relationship. The inverse of cosine similarity is said to be a novelty. Obtained novelty has conceptual divergence, semantic relevance, and contextual similarity. The result showed that hPAM provided more specificity and sensitivity. hPAM provides better accuracy.

B. Document Level Novelty Detection:

Trithankar Ghosal, Tanik Saikh [2], proposed a work which said that with the help of methods of TE, novelty can be determined. The supervised machine learning approach is used in the TE system. Similarity matrices are used by this approach. The incoming document is compared to the documents which are already seen by the system for novelty. Features of TE are used for detecting novelty. The TE system is based on machine learning which takes similarity measures as features. Features may include different types such as cosine, dice which are vector-based or Jaccard, overlap or harmonic which are set based. A system is provided with a number of documents. The target documents are compared with system documents to check whether it can be determined by system document or not. If the target document can be fully retrieved from a single system document or from all documents of the system then it is treated as non-novel. If the target document contains new data which can't be determined by any of the document of the system then it is treated like a novel. The sparsity led to the development of document level novelty detection corpus. The topic around 202 of politics and business domain are taken in the corpus. There exist three source documents and at least one novel and non-novel document in each topic. Three sources of documents are compared with an incoming document of the same topic. Similarity measures are used for the computed similarity score between the incoming document and three source documents. Three similarity sources are obtained.

To arrive final measure we use two methods:
Maximum: Three source documents are compared to the target document. Using similarity measure three sources are computed. Out of all the three sources, the maximum is chosen. We assume that the similarity score of the non-novel document will be highest. Whereas the novel document will be lexically far away in distance and will contain new information. For e.g. We take three source documents SS1, SS2, SS3, and a target document St. Three scores ds1,ds2,ds3 are computed for each feature. The value for each feature is considered as the maximum score computed.
Averaging: In this method, the target document is compared to three source documents using similarity measures. The similarity score is calculated for each source document. There will be three similarity score. In this, the average of the three similarity scores is calculated. The value for each feature will be computed the average score. When the comparison within a novel document is done then the class is labeled as Entailed. When the comparison with a non-novel document then the class is labeled as Non-Entitled. This establishes a relation between TE and novelty.

C. D2S: Document To Sentence Level Framework For Novelty Detection

Flora S Tsai, Vizhang [3] proposed detecting novelty in the document is difficult than detecting novelty in sentences. So, this method was proposed for finding novelty using sentences. Detecting novelty at the sentence level is easier because the document of the new domain will always contain new information. As the document comprises of sentences. If we find novelty using sentence than novelty using documents is automatically determined. For novelty, documents and sentences have always been two different things. In this method, segmentation is performed on documents and documents are segmented into sentences. The similarity is computed between incoming sentences and previously seen sentences. The cosine similarity matrix is used for computing similarity. Cosine similarity is given as:

Cos theta = M.N / ||M||.||N||

Where M and N represent vectors of sentences. For cosine similarity, vectors of sentences are computed.

With the help of computed cosine similarity, Novelty score is computed as :

Novelty Score = min (1-cosine similarity)

This novel score is to be compared with a threshold value. If the score is above the threshold then the sentence is novel otherwise not. It also brought the concept of Novel Rate. Novel Rate is used for finding novel sentences. The novel rate can be defined as a number of identified novel sentences divided by the total number of sentences in a document. This method provided a capability of finding redundant information and novel information in data.

III. FLOW CHART FOR HASHING AND CLUSTERING BASED NOVELTY DETECTION

In our proposed approach we fetch input from user and from the history of the system. The incoming document is compared with the history documents using hashing and clustering technique as stated in Algorithm 1. If the input source is found with new and relevant information then it is marked as novel otherwise non- novel. The flow chart of steps of technique is represented in figure 1.

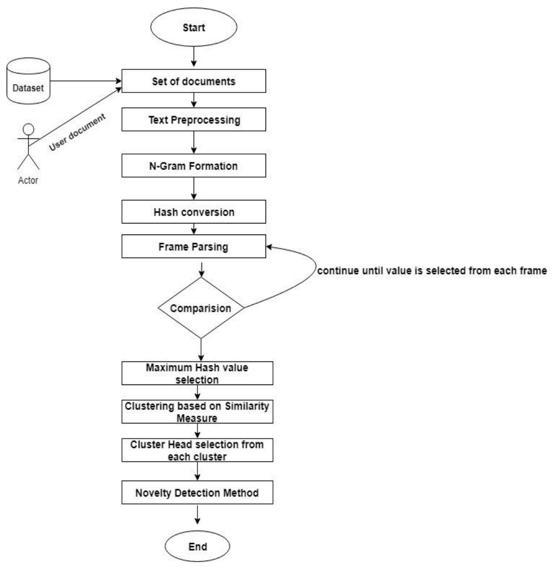

Start
Dataset
Actor
User document
Set of documents
Text Preprocessing
N-Gram Formation
Hash conversion
Frame Parsing
continue until value is selected from each frame
Comparision
Maximum Hash value selection
Clustering based on Similarity Measure
Cluster Head selection from each cluster
Novelty Detection Method
End

Figure 1Flow Work for Hashing and Clustering based Novelty Detection Technique

IV. PROPOSED ALGORITHM

A. Steps of Hashing and Clustering Based Novelty Detection Technique

a). Text Pre-processing

Documents may contain different formats and languages. Text Preprocessing is a process of conversing the data in to a analyzable and useful form for the approach we are using. It is used for finding useful information from unstructured data. Preprocessing plays an important role in text mining. When a user query is submitted for some information retrieval. Documents are retrieved against query. Before retrieval the documents has to go through text preprocessing phase. Text preprocessing also helps in reducing the size of data set. The words which are not useful are removed by text preprocessing.

```
Algorithm 1: String Tokenizer
Input:all_files[] (containing user file and all documents files)
Output: tokens[] of each file
Begin
    1.   for each line in all_files[k]
         1.1.   For each word[i] in line
                1.1.1.  if (word[i] == ' ')
                             Token[k] = append(word[i])
                1.1.2.  else
                             continue
         1.2.  end for
    2.   return Token[k]
    3.   end for
```

1) Tokenization

Tokenization is a process of dividing the sentences of documents in to smaller strings or tokens as shown in algorithm 1. Documents can be broken down to sentences, and sentences can be broken down to strings. These strings are known as tokens.The sentences are divided into different parts and each part act as an individual for next process.

```
Algorithm 2: Removal of stopwords
Input: stopwords[] ,flag=1
Output: Tokens_without_stopwords[]
Begin
    1.   For every tokens in all_files[k]
         1.1.  s1 = token[k]
         1.2.  s1 = s1. toLowerCase();
         1.3.  for( i=0; i< Stopwords.length;i++)
               1.3.1.  if( s1.equals(stopwords[i])
                            flag =0;
         1.4.  if(flag!= 0)
               1.4.1.  print( s1)
         1.5.  End for
    2.   End for
```

2) *StopWords Removal*

There are many words which occur in documents repeatedly and are basically used for merging two sentences. These words are basically known as stopwords. They are extra words which are not useful in content of documents. As they are present repeatedly they make the document more complex and difficult to understand. Words like a, an, the, are the examples of stopwords. It is necessary to remove the stopwords from the document. They make the document look heavier and make difficult for further processing. Articles, prepositions are said to be stopwords. A file of stopwords is provided which are to be removed from text file. Each word of text file is read and the listed stopwords are replaced by empty strings as shown in algorithm 2. Search Engines removes stopwords and saves time and space.

3) *LowerCasing*

It is the easiest form of text pre-processing. Input capitalization may lead to different outputs. In this words with different cases are converted in to lowercases. In this all the strings are transformed to lowercases. All the input is of same format. In java, String toLowerCase () method is used for transforming all the characters of strings in to lowercase as shown in algorithm 2

b). *N-Gram formation*

N-Gram formation is a process of converting string in to substring. N is used for representing a number. N tells how many words will be chosen in 1 gram. The input for N-Gram is a preprocessed string and a value of N. N can be 1, 2,.....,n depending upon user's requirement as shown in algorithm 3. If we take N=1 the N-Grams formed are known as unigrams. If we take N= 2 then the grams formed are bigrams. If we take N=3 then grams formed are trigrams and so on. For example, we are given with a string S.

S = "House was cleaned by sita." N=1, 2, 3.

Unigrams formed are: - 'House' 'was' 'cleaned' 'by' 'sita'.

Algorithm 3: generate N-grams
Input: length of n-gram(n), str= token[]
Output: N-grams[]
Begin

```
1.   if (str.length() < n )
     1.1. return;
2.   if (str.length() == n )
     2.1.  int counter = 0;
     2.2.  while (counter < n)
           2.2.1. nGrams.add(str.substring(counter));
           2.2.2. counter++;
     2.3.  end while
3.   end if
4.   int counter = 0;
     4.1.  String gram = "";
     4.2.  while (counter < n)
           4.2.1. gram += str.charAt(counter);
           4.2.2. counter++;
           4.2.3. nGrams.add(gram);
           4.2.4. generateNGrams(str.substring(1), n);
     4.3.  end while
```

The value of N can vary from user to user. N-grams are consecutive sequence of strings. N-grams are N character slice of a string. They can be evaluated as-

$$N= (p-m+1)$$

p represents number of letters in document and m represents size of N-grams. N-grams are generated from tokens after removal of spaces as shown in figure 2. The size of N=5.

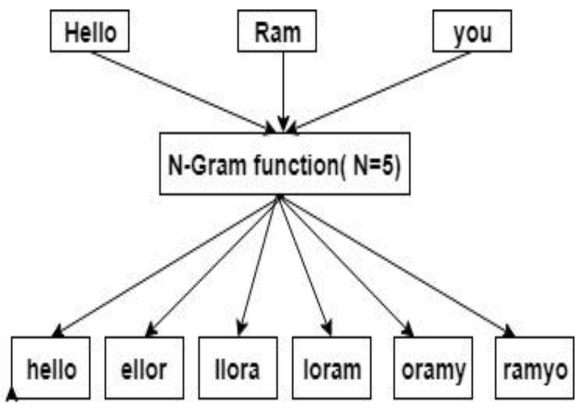


Figure 2 N-gram Formation

Algorithm 4: Forming of hashes
Input: n-grams[], MD5
Output: hashtext[]
Begin

```
1. Call the MEssageDigest instance
2. Byte[] input-= n-grams.getBytes();
3. Biginteger number = new Biginteger(1,input)
4. String hashtext = number.toString(40);
5. while (hashtext .length()< 32)
   5.1. Hashtext = "0"+ hashtext;
6. End while
7. Return hashtext.substring(0,5).
```

C. *Hash Conversion*

Each character of gram is represented by an ASCII value. It converts grams in to corresponding hash value. Each character is converted to ASCII value. Hashing is a process of conversion of grams in to short fixed length value as shown in algorithm 4 .It is performed because it is easy to find short length value than to find original string. The search process will involve formation of hash value and then using it to find match for a given value. Hashes are formed to avoid overwhelming computations. So we require a part of n grams to be used for comparison and the n-grams are converted to hash values.

The equation for hash formation can be given by

$H(d_k) = d1*m\text{^}(k-1)+d2*m\text{^}(k-2) +...+d(k-1)*m+ d\text{^}k$

d: ASCII character
m: basis of primes
k: value of k-grams.

The input to hash function can be arbitrary but the output is fixed referred as n bit output. This process is hashing of data. The converted small values are known as Message Digest as shown in figure.

As the output is a hexadecimal value. A parseInt function of decimal format class is used for converting it to decimal values. A number of hashes are made for each document corresponding to each n-gram of document.

D. Frame Parsing

It is method of converting hashes in to frames. The input provided contains two parameters. First parameter is the hash value or message digest which is the output of MD5 method. The second parameter is f which tells number of hashes to be kept in frame. In this work we have taken f as '4'. This parsing is done to ensure that a minimum value is always available for selection from each frame. A function substring is used for providing the value of 'f'. The frames are created according to size of f. The output is stored in a array list. Each frame will contain a value which will be used for comparison of two documents. Each frame will contain equal number of hashes in it.

E. Minimum Hash Value Selection

In previous phase, frames of equal size were formed. Each frame contains equal number of hashes. For further processing of data we need to choose minimum value from each frame. All the values in a frame are compared to each other in order to find least value. The value which is selected from each frame is minimum hash value.

The reason of choosing the value as minimum is that the minimum value in one frame is likely to be the minimum value in other frames too. It is said that minimum of 'f' random number is smaller than one additional random number. The number of minimum values selected is much less as compared to number of frames. This makes the document to be represented by small number of values. This provides scalability to documents. When there are two similar hash values in two frames then the value in rightmost frame is chosen to be the minimum hash value. This all selected minimum hash values together represent a document.

```
Algorithm 5: Clustering and novelty
Input: min_hashes[]
Output: Novelty percentage
Begin
    1.   For each document k in database
         1.1.  For all i in min_hashes[]
               1.1.1. Sum = sum+ min_hashes[i]
         1.2.  End for
    2.   Centroid = sum / (i-1)
    3.   For all i in min_hashes
         3.1.  Result = result + (centroid – min_hashes[i]) * (centroid- min_hashes[i])
    4.   End for
    5.   Euclidean distance [k]= sqrt(result)
    6.   For each document k
         6.1.  If (Euclidean _distance[k]<= 30)
               6.1.1. Set it in cluster 1[]
         6.2.  Else If (Euclidean _distance[k]<= 60)
               6.2.1. Set it in cluster 2[]
         6.3.  Else (Euclidean _distance[k]<= 100)
               6.3.1. Set it in cluster 3[]
         6.4.  End if
    7.   End for
    8.   Find min_distance in each cluster
         8.1.  Make cluster_head[k] = min(euclidean _distance[k])
         8.2.  Similarity= jaccard[ (user(min-hashes[]) / cluster_ head(min_hashes[])
         8.3.  Novelty percentage = 1- similarity
```

F. Clustering based on Similarity Measures

Clustering is a method of grouping objects together. The objects are placed in some groups. These groups are generally called as clusters. The criteria for division of groups can vary from algorithm to algorithm. The clusters are majorly vast and dense. The criteria for division of groups can be based on properties like threshold, distance, similarity measurement depending upon set of data. It can be considered as an iterative process which continues until desirable results are achieved. Similarity based clustering has been considered a powerful technique. The principle of similarity based clustering is that similar objects are placed together in same cluster. In this we assume that data points which are closer to each other are similar than the documents which are far away. Here we consider distance calculation as a similarity measure for clustering to form clusters. The distance for each document is calculated from centroid. The centroid is chosen as average sum of minimum hash values selected. The distance of documents is compared to threshold value. The

Documents with nearby distance are grouped together. The documents which are far away are placed in another cluster. The documents are divided in to clusters on the basis of similarity measures. Values to be taken for threshold can vary from user to user.

G. Cluster Head Selection

Cluster head is selected on basis of similarity function computed in previous phase. Similarity Measure is calculated for each document. Clustering is performed on documents as seen in previous phase. Similarity of all the documents with in a cluster is compared. The document with the least similarity measure will be chosen as cluster head in each cluster. The similarity function to be used is Euclidian distance measure. A cluster may contain number of documents but it will have a single cluster head for each cluster. The cluster head will represent the whole cluster. From each cluster a single cluster head is chosen on basis of distance measure.

H. Novelty Detection

This the final phase of this algorithm. The minimum hash values of user are compared with the minimum hash values of cluster head in order to find novelty detection. Values are compared with each cluster head and similarity between user and each cluster head is determined. The similarity coefficient used is Jaccard coffiecient. It is calculated as intersection over union of values.

Jaccard (user , cluster head)= hash value of user hash value of cluster head / hash value of user U hash value of cluster head

Intersection – ∩

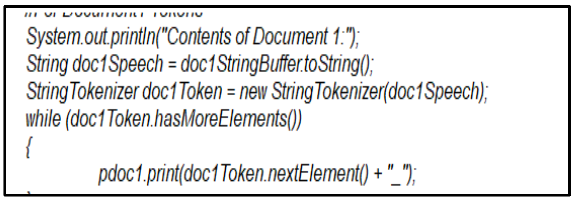
Union – U

After calculation of similarity between user and cluster head novelty is calculated. Novelty is defined as 1- similarity as shown in algorithm 5. With each cluster head novelty of user is calculated. novelty is calculated for user document. This is the last phase for novelty detection.

V. IMPLEMENTATION

- Upload documents stored in the system for computation purpose
- Provide a user`s document which is to be compared with documents of system
- Perform tokenization on documents to break down the sentences of documents in to tokens.

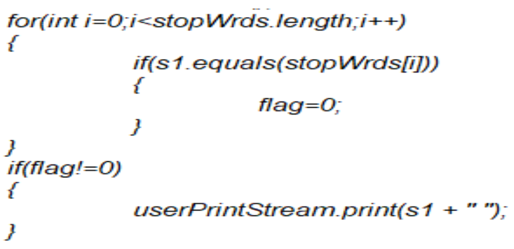
Java provides readymade class and method for tokenization. StringTokenizer is a method used for dividing the sentences in to tokens as shown in figure 4. In this a delimiter can be used for delimiting the strings .Default delimiters are also considered.



```
System.out.println("Contents of Document 1:");
String doc1Speech = doc1StringBuffer.toString();
StringTokenizer doc1Token = new StringTokenizer(doc1Speech);
while (doc1Token.hasMoreElements())
{
        pdoc1.print(doc1Token.nextElement() + "_");
```

Figure 4 Tokenization of sentences of documents.

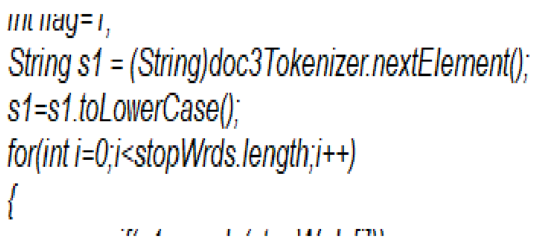
The impurities are removed from documents. Removing impurities implies removal of stopwords and converting all the characters of tokens in to lowercase In java, File of stopwords is provided which are to be removed from document. Each word of text file is read and the stopwords are replaced by empty strings as shown in figure 5.



```
for(int i=0;i<stopWrds.length;i++)
{
            if(s1.equals(stopWrds[i]))
            {
                        flag=0;
            }
}
if(flag!=0)
{
            userPrintStream.print(s1 + " ");
}
```

Figure 5 Removal of stopwords

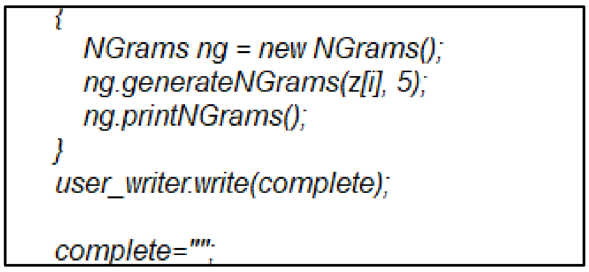
After removal all the characters are transformed in to lowercase. In java, String tolowerCase method is used for transforming in to lowercase as shown in figure 6.



```
int flag=1;
String s1 = (String)doc3Tokenizer.nextElement();
s1=s1.toLowerCase();
for(int i=0;i<stopWrds.length;i++)
{
```

Figure 6 lowercase transformation

After removal of impurities N-grams are generated corresponding to each token. In java, generateNGrams function is used for generating n-grams as shown in figure 7. As an input a string and value of n is provided.

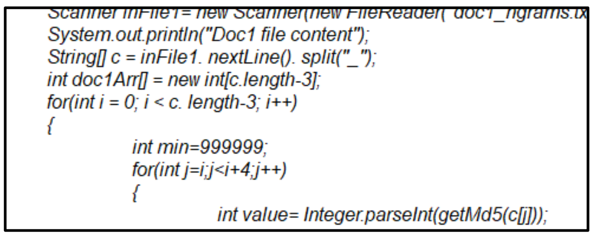


```
{
    NGrams ng = new NGrams();
    ng.generateNGrams(z[i], 5);
    ng.printNGrams();
}
user_writer.write(complete);

complete="";
```

Figure 7 Generation of N- grams of size N=5.

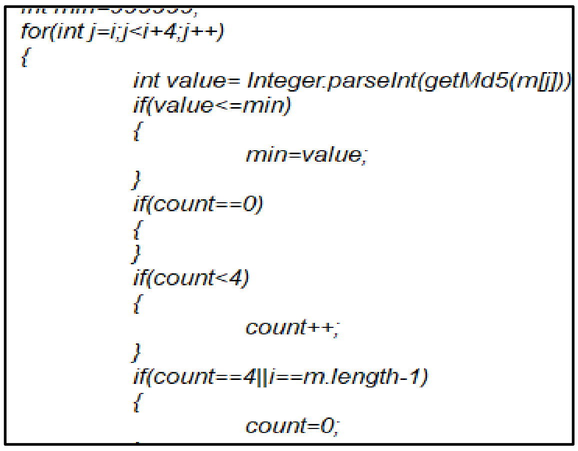
N-grams are generated. Hashes are formed with respect to every N-gram. It provides a numeric value corresponding to each N-gram. In java, MD5 class is used and getInstance method is used for formation of hashes as shown in figure 8. The output provided is in hexadecimal format. So a function parseInt of class decimalFormat is used for converting it to decimal values.



```
Scanner inFile1= new Scanner(new FileReader( doc1_ngrams.tx
System.out.println("Doc1 file content");
String[] c = inFile1. nextLine(). split("_");
int doc1Arr[] = new int[c.length-3];
for(int i = 0; i < c. length-3; i++)
{
        int min=999999;
        for(int j=i;j<i+4;j++)
        {
                int value= Integer.parseInt(getMd5(c[j]));
```

Figure 8 Formation of hashes

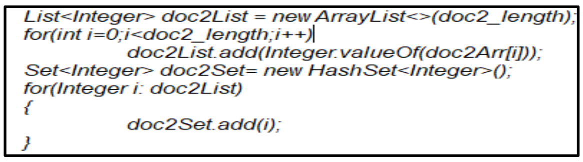
After this frames are formed of fixed size. Each frame contains equal number of hashes. Frames are formed of size 4 as shown in figure 9.



```
int min=999999;
for(int j=i;j<i+4;j++)
{
        int value= Integer.parseInt(getMd5(m[j]))
        if(value<=min)
        {
                min=value;
        }
        if(count==0)
        {
        }
        if(count<4)
        {
                count++;
        }
        if(count==4||i==m.length-1)
        {
                count=0;
```

Figure 9 Formation of frames of size=4

After formation of frames a minimum hash value from each frame is selected. Each frame contains equal number of hashes. All the hashes of a frame are compared to each other and the minimum hash value is chosen from each frame. For each document an arrayList is created which contains set of all the minimum hash values as shown in figure 10.



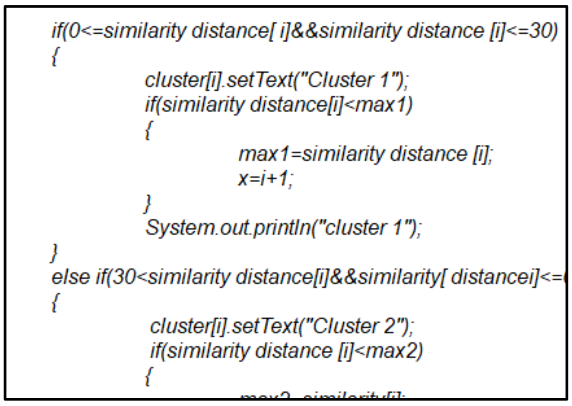
```
List<Integer> doc2List = new ArrayList<>(doc2_length);
for(int i=0;i<doc2_length;i++)
        doc2List.add(Integer.valueOf(doc2Arr[i]));
Set<Integer> doc2Set= new HashSet<Integer>();
for(Integer i: doc2List)
{
        doc2Set.add(i);
}
```

Figure 10Minimum Hash value selection

Selection of minimum hash value a formation of set which contains all minimum hash values for a document

For each document stored in system, Euclidean distance is computed as shown in figure 11. Similarity distance is computed for each document.

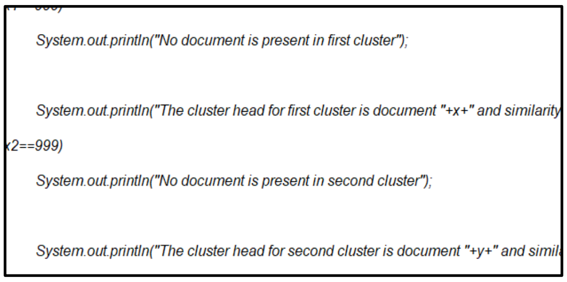
Clustering is performed on the basis of similarity distance as shown in figure 12. The distances computed for each document is compared to threshold values. The documents which are near to each other will be placed in a single cluster and the documents which are far away are placed in another cluster.



```
if(0<=similarity distance[ i]&&similarity distance [i]<=30)
{
        cluster[i].setText("Cluster 1");
        if(similarity distance[i]<max1)
        {
                max1=similarity distance [i];
                x=i+1;
        }
        System.out.println("cluster 1");
}
else if(30<similarity distance[i]&&similarity[ distancei]<=
{
        cluster[i].setText("Cluster 2");
        if(similarity distance [i]<max2)
        {
```

Figure 12 Clusters are formed on basis of similarity distance.

From each cluster a cluster head is selected as shown in figure 13. In each cluster the document with minimum distance is selected as cluster head.



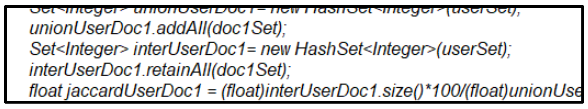
```
        System.out.println("No document is present in first cluster");


        System.out.println("The cluster head for first cluster is document "+x+" and similarity
x2==999)
        System.out.println("No document is present in second cluster");


        System.out.println("The cluster head for second cluster is document "+y+" and simil
```

Figure 13 Selection of cluster head from each document.

After selection of cluster head, similarity is computed between minimum hashes of user and minimum hashes of cluster head as shown in figure 14.



```
Set<Integer> unionUserDoc1= new HashSet<Integer>(userSet);
unionUserDoc1.addAll(doc1Set);
Set<Integer> interUserDoc1= new HashSet<Integer>(userSet);
interUserDoc1.retainAll(doc1Set);
float jaccardUserDoc1 = (float)interUserDoc1.size()*100/(float)unionUse
```

Figure 14 Similarity computation using Jaccard

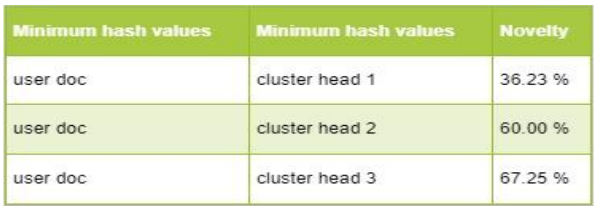
VI. RESULTS

A. Hash Formation

N-grams	Hashes
Featu	385
Eatur	124
Ature	246
Turei	324
Urein	229
Reind	146
Eindi	310
India	212

Table1 hash formation

Table 1 shows formation of hashes. Corresponding to each n-gram a hash value is formed.These hashes represent the document as a whole. For comparison process hashes are required. The size of hashes is chosen by user. Here we have taken size of 3. Each n-gram will have different hash value unless it is similar to another n-gram.



Minimum hash values	Minimum hash values	Novelty
user doc	cluster head 1	36.23 %
user doc	cluster head 2	60.00 %
user doc	cluster head 3	67.25 %

Table 2 Novelty Detection

Table 2 shows computation of Novelty Detection using Jaccard coefficient. The minimum hash values of user`s document are compared with the minimum hash values of the document in the cluster head. A similarity coefficient like Jaccard is used for computing similarity and from similarity novelty is computed.

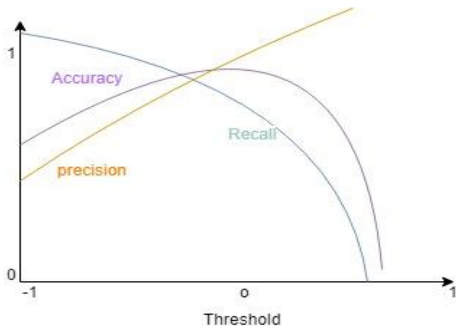


Figure 15 Precision, Recall and Accuracy graph

Figure 15 shows improvement in accuracy with the use of hashing and clustering based novelty detection technique. Accuracy is improved and the complexity is reduced. Precision is defined as number of correct results divided by number of all returned results. Recall is the number of correct results divided by number of results that should have been returned. Accuracy is correctly predicted observation to the total observations.

VII. CONCLUSION AND FUTURE SCOPE

The main motive of this work was to provide a method which provides appropriate and novel information to the user as fast as possible with improved accuracy. In this work, Novelty Detection was presented with a new definition. It is a method of not only discovering novel information but relevant information also. Based on this definition, we have proposed hashing and clustering based novelty detection methods. The evaluation result shows this technique produces Novelty Detection. This method has advanced efficiency as shown in the graph. This method has reduced time complexity as the user document does not have to be matched with all documents in history. In the future, more improvement can be done. Accuracy can be increased. There can be various methods used for choosing the cluster method. Text summarization can be used for cluster head solution. In this, the review of all documents of a cluster can be interpreted as cluster head. Many More approaches can be used.

REFERENCES

[1] Sendhilkumar, Nachiyar S Nandhini " Novelty Detection via Topic Modeling ", Department of Information science and Technology ,Anna University, Chennai, Tamil Nadu

[2] Tirthankar Ghosal , Amrita Salam" A Corpus for Document Level Novelty Detection", Indian Institute of technology Patna Bihta,Bihar.

[3] Agus T. Kwee, Flora S. Tsai " Sentence Level Novelty Detection in English and Malay" Nayang Technological University, School of Electrical and Electronic Engineering, Singapore.

[4] Yi Zhang, Flora S. Tsai . Combining Named Entities and Tags for Novel Sentence Detection . Nanyang Technological University 50 Nanyang Avenue Singapore 639798 .

[5] Yiming Yang, Jian Zhang, Jaime Carbonell, Chun Jin . Topic-conditioned Novelty Detection. School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213-8213, USA .

[6] Marek Gajewski, Janusz Kacprzyk, Sławomir Zadrożny. TOPIC DETECTION AND TRACKING: A FOCUSED SURVEY AND A NEW VARIANT. Systems Research Institute, Polish Academy of Sciences, Warszawa, ul. Newelska 6, 01-447 Warszawa, Poland Warsaw School of Information Technology, ul. Newelska 6, 01-447 Warszawa, Poland .

[7] Xiaoyan Li , W. Bruce Croft . Novelty Detection Based on Sentence Level Patterns . Center for Intelligent Information Retrieval Department of Computer Science University of Massachusetts, Amherst MA 01003.

[8] Michael Gamon . Graph-Based Text Representation for Novelty Detection . Microsoft ResearchRedmond, WA 98052

[9] Ming-Feng Tsai, Ming-Hung Hsu, and Hsin-Hsi Chen . Similarity Computation in Novelty Detection. Department of Computer Science and Information Engineering National Taiwan University 1, Section 4, Roosevelt Road, Taipei, Taiwan, 106 .

[10] Saul Schleimer , Daniel S. Wilkerson, Alex Aiken . Winnowing: Local Algorithms for Document Fingerprinting. University of Illinois, Chicago, Computer Science Division UC Berkeley.

[11] Anton Yudhana, Sunardi , Iif Alfiatul Mukaromah . Implementation of Winnowing Algorithm with Dictionary English-Indonesia Technique to Detect Plagiarism . Department of Electrical Engineering Universitas Ahmad Dahlan Yogyakarta, Indonesia, Master of Informatics Engineering Universitas Ahmad Dahlan Yogyakarta, Indonesia .

[12] Norzima Elbegbayan . Winnowing, a Document Fingerprinting Algorithm . Department of Computer Science Linkoping University .

[13] Rhio Sutoyo , Insan Ramadhani, Angger Dwi Ardiantma , Sanditya Cakti Bavana . Detecting documents plagiarism using winnowing algorithm and k-gram method . IEEE Conference, 2017.

[14] Agung Toto Wibowo, Kadak W. Sudarmadi , Ari M. Barmawi . Comparison between fingerprint and winnowing algorithm to detect plagiarism fraud on Bahasa Indonesia documents . IEEE Conference, 2013.

[15] Dubravko. Milijkovic . Review of novelty detection methods. 33rd International Conference .

[16] Marco A. F. Pimentel ,David A. Clifton , Lei Clifton ,Lionel Tarassenko . A review of novelty detection. Journal Signal Processing , Volume 99, 2014.

[17] Markos Markou and Sameer Singh. Novelty Detection: A Review Part 1: Statistical Approaches. PANN Research, Department of Computer Science University of Exeter, Exeter EX4 4PT, UK .

[18] .A. Clifton, H. Yin, and Y. Zhang, “Support vector machine in novelty detection for multi-channel combustion data,” in Proc. 3rd International Conference Advance Neural Network.-Volume Part III, 2006 ,pp.836–843.

[19] E.J.Spinosa, deA.C.P.L.F.deCarvalho, andJ.Gama, “Cluster- based novel concept detection in data streams applied to intrusion detectionincomputernetworks.”inProc. ACMSymp. Application Computer 2008, pp.976–980.

[20] L. Khan, M. Awad, and B. Thuraisingham, “A new intrusion detection system using support vector machines and hierarchical clustering,”TheVLDBJ.,vol.16,no .4,pp.507–521,2007.