ISSN 2278-3091
Volume 9 No.2, March - April 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse160922020.pdf
https://doi.org/10.30534/ijatcse/2020/160922020

Automatic Summarization and Keyword Extraction from Multiple Wiki Articles and Books

Dr. S. Pitchumani Angayarkanni[1]
[1]Associate Professor, Department of Computer Science, Lady Doak College, Madurai, Tamil Nadu, India,
pitchumca@gmail.com

ABSTRACT

Summarization is the process of creating a brief and accurate overview of a very lengthy document. Automatic text summarization plays a vital role to provide a brief overview of the huge volume of content available online. This will help the users to get relevant information faster and accurate. Various research carried out in this field involves summarization of small documents and less research has been done in the field of lengthy document summarization. The proposed algorithm involves summarization of lengthy documents like books, online journal article contents and medical records in an effective, accurate and less time consumption. Natural Language Processing(NLP) technique is implemented in the proposed Deep learning algorithm for automatic text summarization. The Recurrent Neural Network-based approach is used to perform the above process. The performance analysis of the algorithm clearly shows that it can summarize very long documents of more than 500 pages to 1000 pages in 2ms with high accuracy of 98.94%.

Key words: Natural Language Processing, Summarization, Deep Learning Algorithm, Recurrent Neural Network, Encoder, Decoder, and Performance Measure.

1. INTRODUCTION

A huge volume of documents available on the Internet with vital information. There is a need to extract useful and valid information from these sources. Therefore, we need an automated system to extract relevant information from these sources. The text mining concept is applied to extract large quantities of text to derive high-quality information. Different phases involved in Text Summarization are topic detection, interpretation, and summarization. The basic framework involves keyword modeling, sentence-word, tokenizer, sentiment intensity analyzer and finding the polarity score of the summarized sentence. There are two types of summarization methods abstractive and extractive methods. Abstractive summarization generates meaningful new phrases and sentences from the source documents. The extractive technique involves the selection of sentences and phrases from the source document. The proposed method involves an abstractive summarization approach with deep learning RNN algorithm for automatic text summarization. The proposed technique is used to merge and summarize the Wikipedia articles from the URL given as input from the user using a web scraping technique. The proposed algorithm involves the following phases which are explained in detail in the following sections of the paper. Section 1: a general framework of the proposed technique with extract text, split sentence, the formation of word
embedding using vectors, evolving similarity matrix, graph construction and sentence ranking information of summary. Section 2: Implementation of Encode-Decoder based RNN for automatic text summarization, Section 3: Performance evaluation and validation.
Challenges involved in the Natural Language Processing for automatic summarization involves difficulty in extraction of meaningful, evaluation of summary and timely summarization techniques. There is a need to automatically extract meaningful content from the information sources. There is a need for a novel approach to overcome these challenges. The resultant summary generated through the proposed method will help the users to reduce reading time, research for information and enhance the amount of information required. Two types of summarization technique exist which includes extraction-based and abstraction based summarization. The proposed technique involves deep learning-based abstractive summarization of information from multiple Wikipedia articles on the same topic.
copyright form and the form should accompany your final submission.

2. PROPOSED ARCHITECTURE

Colin Raffel et. al. (2019) proposed a transfer learning technique with a unified text to text transformer for efficient text summarization, question answering and text classification with an accuracy of 94% to 95%. Figure 1.1 shows the methodology proposed for the study. The input Wikipedia articles URLs are provided as input by the user. The content in the articles are retrieved using web scraping techniques and the resultant documents are merged. The preprocessing step involves converting the entire content to lowercase, split the paragraphs into sentences, stemming, lemmatizing, stop word removal, text normalization, and Text augmentation. Stemming is defined as a heuristic approach that is used to reduce inflection in words (e.g. troubled, troubles) to their root form (e.g. trouble). The root represents a canonical form of the original word. Porters stemming algorithm is used for the stemming approach. Vectorization is defined as "The

process of converting **NLP** text into numbers is called **vectorization** in Machine Learning". Different ways to convert text into vectors are: Counting the number of times each word appears in a document. Calculating the frequency that each word appears in a document out of all the words in the document [1]. The sentiment Analysis technique is implemented using Valence Aware Dictionary and Sentiment Reasoner (VADAR) in python. It is a rule-based method of sentiment analysis. It gives positivity and negativity score in the range of -1 to 1.

The Bag-of-Words technique is used to extract the features from the text using the Machine Learning algorithm. The frequency of token is estimated from the tokenization technique. Each sentence is treated as a separate document and a vector is created. The vector converts text in the form that can be used by the machine learning algorithm. CountVectorizer method is applied on Terms Frequency, i.e. counting the occurrences of tokens and building a sparse matrix of documents x tokens. Term Frequency and Inverse document frequency is calculated. The resultant Bag-of-words generated is given as input to RNN or Long Short Term Memory LSTM which is represented as encoder and decoder components. This involves 2 phases the Training and Inference Phase respectively.

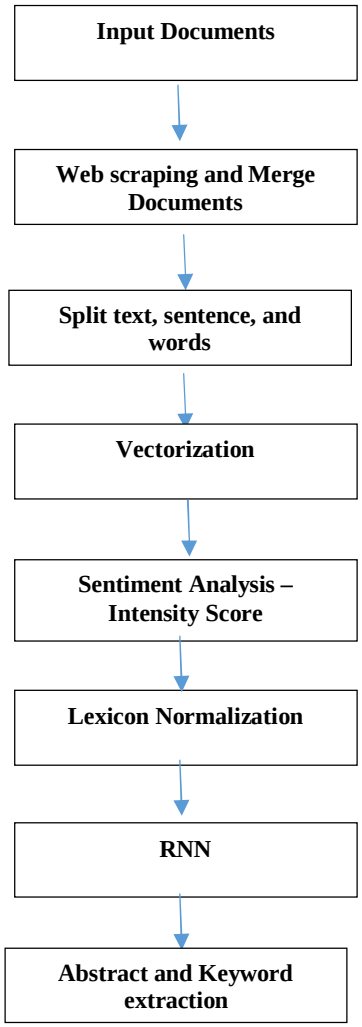


Figure 1: Proposed Framework

Training Phase:

Train the model to predict the target sequence offset. This is done by a one-time step.

Encoder: Encoder LSTM reads the given input sequence at each time step one word is fed into the encoder. The encoder processes the information and captures the contextual information at every time step.

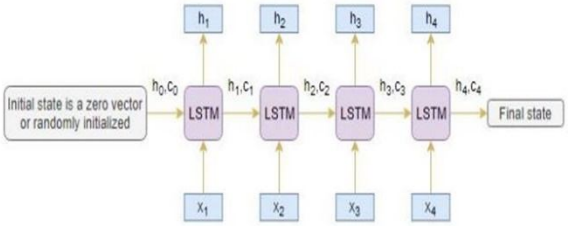


Figure 2: Encoder

Figure 2 indicates the hidden state (h_i) and cell state (c_i) of the last time step are used to initialize the decoder

Decoder: Reads the target sequence word-by-word and predict the sequence offset in a one-time step. It is trained to predict the next word in the sequence given the previous word.

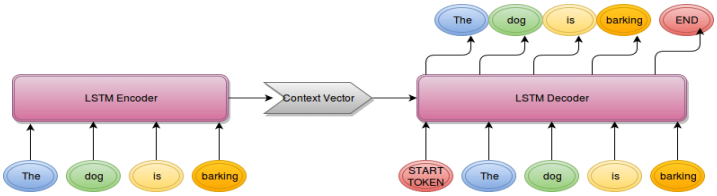


Figure 3: Decoder

Figure 3 indicates <start> and <**end**>are the special tokens which are added to the target sequence before feeding it into the decoder". The target sequence is unknown while decoding the test sequence. We predict target sequence by sending the first word to the decoder as <start> token. The <**end**>token signals the end of the sentence.

Inference Phase:

After the training phase, the model is tested on a new set of sequences for which the target is unknown. Figure 4 shows the architecture of the Inference Phase followed by the pseudocode in Figure 5.

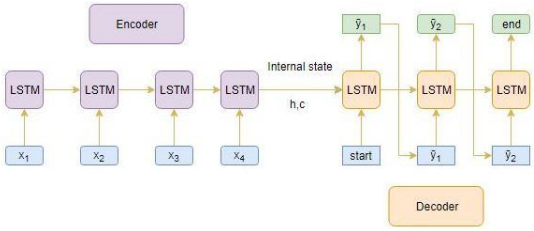


Figure 4: Inference Phase

steps to decode the test sequence:

A. Encode the entire input sequence and initialize the decoder with internal states of the encoder
B. Pass <start> token as an input to the decoder
C. Run the decoder for one timestep with the internal states
D. The output will be the probability for the next word. The word with the maximum probability will be selected
E. Pass the sampled word as an input to the decoder in the next timestep and update the internal states with the current time step
F. Repeat steps 3 – 5 until we generate <end> token or hit the maximum length of the target sequence

Figure 5: Pseudocode for Inference Phase

2.1 Implementation

Step 1: Input the keyword and apply web scraping

Web scraping technique is used to fetch the huge volume of data from the Internet. The python library used for web scraping techniques are requests and Beautiful Soup for scraping and parsing data from the Web as indicated in figure 6.

```
                    film_titles.append(link['title'])
                    # get the link and add to that list
                    film_links.append(link['href'])
    #be a conscientious scraper and pause between scrapes
    time.sleep(1)
print('Number of Film Links Collected:', len(film_links))
print('Number of Film Titles Collected:', len(film_titles))
# remove film links that don't have a description page on Wikipedia
new_film_links = [i for i in film_links if 'redlink' not in i]
# same goes for titles
new_film_titles = [i for i in film_titles if '(page does not exist)' not in i]
print('Number of Film Links with Wikipedia Pages:',len(new_film_links))
print('Number of Film Titles with Wikipedia Pages:',len(new_film_titles))
#use this list to fetch from the API
title_links = list(zip(new_film_titles, new_film_links))
```

```
('Collecting films from decade', 'https://en.wikipedia.org#See_also')
('Collecting films from decade', 'https://en.wikipedia.org#References')
('Collecting films from decade', 'https://en.wikipedia.org#External_links')
('Collecting films from decade', 'https://en.wikipedia.org/wiki/List_of_science_fiction_films_before_1920')
('Collecting films from decade', 'https://en.wikipedia.org/wiki/List_of_science_fiction_films_of_the_1920s')
('Collecting films from decade', 'https://en.wikipedia.org/wiki/List_of_science_fiction_films_of_the_1930s')
('Collecting films from decade', 'https://en.wikipedia.org/wiki/List_of_science_fiction_films_of_the_1940s')
('Collecting films from decade', 'https://en.wikipedia.org/wiki/List_of_science_fiction_films_of_the_1950s')
('Collecting films from decade', 'https://en.wikipedia.org/wiki/List_of_science_fiction_films_of_the_1960s')
('Collecting films from decade', 'https://en.wikipedia.org/wiki/List_of_science_fiction_films_of_the_1970s')
('Number of Film Links Collected:', 737)
('Number of Film Titles Collected:', 737)
('Number of Film Links with Wikipedia Pages:', 708)
('Number of Film Titles with Wikipedia Pages:', 708)
```

Figure 6: Input Source

Step 2: Find the Readability Score of the document

The readability score depends on the complexity and vocabulary of the contents in the document. The words to paraphrase the document is evaluated using Readability score methods. There are various types of formulas used to find the readability score which includes the Dale–Chall formula, the Gunning fog formula, Fry readability graph, McLaughlin's SMOG formula, the FORCAST formula, Readability and newspaper readership, and Flesch Scores. The SMOG formula is used to check the readability score of the words collected from various articles on a similar concept.
Grade Level =1.0430X √(No. of Polysyllabic wordsX(30/No. of Sentences))+3.1291

The proposed readability score source code is specified in figure 7.

```
#from readability import Readability

num_words = 0

with open('../input/paper.txt',encoding='utf-8',errors='ignore') as f:
    for text in f:
        words = text.split()
        num_words += len(words)
        sentence = text.count('.') + text.count('!') + text.count(';') +
text.count(':') + text.count('?')
        words = len(text.split())
        syllable = 0
        for word in text.split():
            for vowel in ['a','e','i','o','u']:
                syllable  = syllable + word.count(vowel)
            for ending in ['es','ed','e']:
                if word.endswith(ending):
                    syllable  = syllable - 1
                if word.endswith('le'):
                    syllable  = syllable + 1
        G = round((0.39*words)/sentence+ (11.8*syllable)/words-15.59)
        print(G)
        if G>=0 and G<=30:
            print ('The Readability level is College')
        if G>=50 and G<=60:
            print ('The Readability level is High School')
        elif G>=90 and G<=100:
             print ('The Readability level is fourth grade')
             print ('This text has %d words' %(words))
        #elif UserFileName not in listOfdir:
         #   print ('This text file does not exist in current directory')
        #elif not(UserFileName.endswith('.txt')):
         #   print ('This is not a text file.')


16
The Readability level is College
```

Figure 7: Readability Score Calculation

Step 3: Text Pre-processing

The tokenizer is used to retrieve the text from the documents. The retrieved text or sentence is split into words using Regular expression tokenizer. The features are extracted from the text using CountVectorizer. The frequency of occurrence of the text is calculated as indicated in figure 8.

```
from nltk.tokenize import sent_tokenize
tokenized_text=sent_tokenize(text)
print(tokenized_text)
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
print(nltk.pos_tag(tokenized_word))
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer
#tokenizer to remove unwanted elements from out data like symbols and
numbers
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer(lowercase=True,stop_words='english',ngram_range =
(1,1),tokenizer = token.tokenize)
text_counts= cv.fit_transform(tokenized_word)
print(text_counts)
['Prediction of kidney function and chronic kidney disease (CKD) through
kidney ultrasound imaging has long been considered desirable in clinical
practice because of its safety, convenience, and affordability.', 'However,
this highly desirable approach is beyond the capability of human vision.',
'We developed a deep learning approach for automatically determining the
estimated glomerular filtration rate (eGFR) and CKD status.', 'We exploited
the transfer learning technique, integrating the powerful ResNet model
pretrained on an ImageNet dataset in our neural network architecture, to
predict kidney function based on 4,505 kidney ultrasound images labeled
using eGFRs derived from serum creatinine concentrations.', 'To further
extract the information from ultrasound images, we leveraged kidney length
annotations to remove the peripheral region of the kidneys and applied
various data augmentation schemes to produce additional data with
variations.', 'Bootstrap aggregation was also applied to avoid overfitting
and improve the models generalization.', 'Moreover, the kidney function
features obtained by our deep neural network were used to identify the CKD
status defined by an eGFR of <60?ml/min/1.73?m2.', 'A Pearson correlation
coefficient of 0.741 indicated the strong relationship between artificial
intelligence (AI)- and creatinine-based GFR estimations.', 'Overall CKD
status classification accuracy of our model was 85.6% higher than that of
experienced nephrologists (60.3%80.1%).', 'Our model is the first
fundamental step toward realizing the potential of transforming kidney
ultrasound imaging into an effective, real-time, distant screening tool.',
'AI-GFR estimation offers the possibility of noninvasive assessment of
kidney function, a key goal of AI-powered functional automation in clinical
practice.']
['Prediction', 'of', 'kidney', 'function', 'and', 'chronic', 'kidney',
'disease', '(', 'CKD', ')', 'through', 'kidney', 'ultrasound', 'imaging',
'has', 'long', 'been', 'considered', 'desirable', 'in', 'clinical',
'practice', 'because', 'of', 'its', 'safety', ',', 'convenience', ',',
'and', 'affordability', '.', 'However', ',', 'this', 'highly', 'desirable',
'approach', 'is', 'beyond', 'the', 'capability', 'of', 'human', 'vision',
'.', 'We', 'developed', 'a', 'deep', 'learning', 'approach', 'for',
'automatically', 'determining', 'the', 'estimated', 'glomerular',
'filtration', 'rate', '(', 'eGFR', ')', 'and', 'CKD', 'status', '.', 'We',
'exploited', 'the', 'transfer', 'learning', 'technique', ',',
'integrating', 'the', 'powerful', 'ResNet', 'model', 'pretrained', 'on',
'an', 'ImageNet', 'dataset', 'in', 'our', 'neural', 'network',
'architecture', ',', 'to', 'predict', 'kidney', 'function', 'based', 'on',
'4,505', 'kidney', 'ultrasound', 'images', 'labeled', 'using', 'eGFRs',
'derived', 'from', 'serum', 'creatinine', 'concentrations', '.', 'To',
'further', 'extract', 'the', 'information', 'from', 'ultrasound', 'images',
',', 'we', 'leveraged', 'kidney', 'length', 'annotations', 'to', 'remove',
'the', 'peripheral', 'region', 'of', 'the', 'kidneys', 'and', 'applied',
'various', 'data', 'augmentation', 'schemes', 'to', 'produce',
```

Figure 8: Tokenizer with Output

Step 4: Sentiment Analysis

The sentiment analysis technique involves analyzing text data and classify the same as negative, positive or neutral polarity. Companies use this technique to analyze the survey responses, review of their products, comments on social media and get valuable information related to their products from the customer's viewpoint.

"Sentiment analysis studies the subjective information in an expression, that is, the opinions, appraisals, emotions, or attitudes towards a topic, person or entity. Expressions can be classified as **positive**, **negative**, or **neutral**". The sentiment analysis pseudocode is specified in figure 9.

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()
for sentence in tokenized_text:
    print(sentence)
    ss = sid.polarity_scores(sentence)
    for k in ss:
        print('{0}: {1}, '.format(k, ss[k]), end='')
    print()
Prediction of kidney function and chronic kidney disease (CKD) through
kidney ultrasound imaging has long been considered desirable in clinical
practice because of its safety, convenience, and affordability.
neg: 0.0, neu: 0.836, pos: 0.164, compound: 0.6249,
However, this highly desirable approach is beyond the capability of human
vision.
neg: 0.0, neu: 0.685, pos: 0.315, compound: 0.5563,
We developed a deep learning approach for automatically determining the
estimated glomerular filtration rate (eGFR) and CKD status.
neg: 0.0, neu: 1.0, pos: 0.0, compound: 0.0,
We exploited the transfer learning technique, integrating the powerful
ResNet model pretrained on an ImageNet dataset in our neural network
architecture, to predict kidney function based on 4,505 kidney ultrasound
images labeled using eGFRs derived from serum creatinine concentrations.
neg: 0.069, neu: 0.833, pos: 0.097, compound: 0.0516,
To further extract the information from ultrasound images, we leveraged
kidney length annotations to remove the peripheral region of the kidneys
and applied various data augmentation schemes to produce additional data
with variations.
neg: 0.0, neu: 1.0, pos: 0.0, compound: 0.0,
Bootstrap aggregation was also applied to avoid overfitting and improve the
models generalization.
neg: 0.137, neu: 0.683, pos: 0.18, compound: 0.1779,
Moreover, the kidney function features obtained by our deep neural network
were used to identify the CKD status defined by an eGFR of
<60?ml/min/1.73?m2.
neg: 0.0, neu: 1.0, pos: 0.0, compound: 0.0,
A Pearson correlation coefficient of 0.741 indicated the strong
relationship between artificial intelligence (AI)- and creatinine-based GFR
estimations.
neg: 0.0, neu: 0.701, pos: 0.299, compound: 0.7506,
Overall CKD status classification accuracy of our model was 85.6% higher
than that of experienced nephrologists (60.3%80.1%).
neg: 0.0, neu: 1.0, pos: 0.0, compound: 0.0,
Our model is the first fundamental step toward realizing the potential of
transforming kidney ultrasound imaging into an effective, real-time,
distant screening tool.
neg: 0.0, neu: 0.876, pos: 0.124, compound: 0.4767,
AI-GFR estimation offers the possibility of noninvasive assessment of
kidney function, a key goal of AI-powered functional automation in clinical
practice.
neg: 0.0, neu: 1.0, pos: 0.0, compound: 0.0,
/opt/conda/lib/python3.6/site-packages/nltk/twitter/__init__.py:20:
UserWarning: The twython library has not been installed. Some functionality
from the twitter package will not be available.
  warnings.warn("The twython library has not been installed. "
```

Figure 9: Sentiment Analysis using VADAR with output

Step 4.1: Frequency Distribution

The frequency of occurrence of the words is determined using the FreqDist function as indicated in figure 10.

```
from nltk.probability import FreqDist
fdist = FreqDist(tokenized_word)
print(fdist)
<FreqDist with 177 samples and 285 outcomes>
fdist.most_common(2)
[('the', 14), ('of', 12)]
```

Figure 10: Probability Distribution

Step 5: Filtered Words

This involves tokenization and filtering the words based on the probability of ranking the words.The frequency of occurrence of filtered words and the pseudocode is explained in figures 11 and 12 respectively.

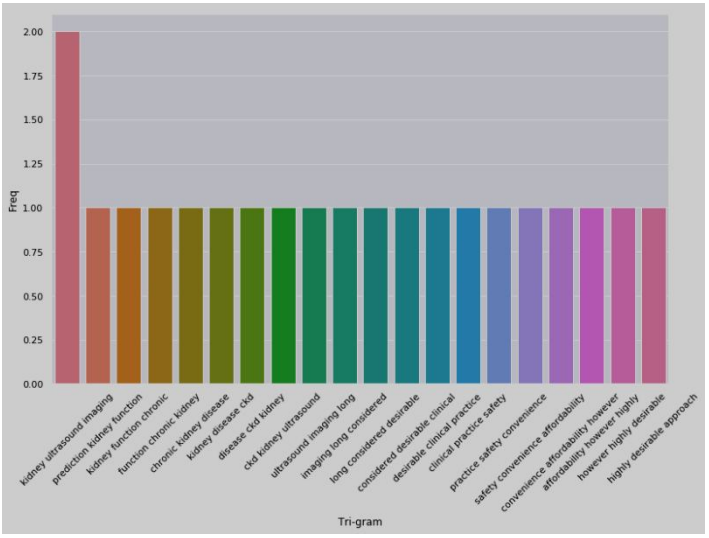


Figure 11: Frequency of Distribution of words

- **Return Sequences = True:** When the return sequences parameter is set to **True**, LSTM produces the hidden state and cell state for every timestep
- **Return State = True:** When return state = **True**, LSTM produces the hidden state and cell state of the last timestep only
- **Initial State:** This is used to initialize the internal states of the LSTM for the first timestep
- **Stacked LSTM:** Stacked LSTM has multiple layers of LSTM stacked on top of each other. This leads to a better representation of the sequence. I encourage you to experiment with the multiple layers of the LSTM stacked on top of each other (it's a great way to learn this)

Figure 12: Pseudocode for the Algorithm

Step 6: Encoding and Decoding

The proposed model involves three basic components: encoder, intermediate encoder vector, and decoder.
The encoder has several recurrent units that accept a single element of the input sequence, retrieve information from the element and move it forward.
Encoder vector contains information of all input for accurate prediction of the decoder. A decoder is a stack with several recurrent units to produce output at a particular time instance. It is a neural machine translation technique. In this technique, one word at a time is provided as input to the encoder with its timestamp. This word is then processed in the LSTM by retrieving the information present in the sequenced input. It is followed by a decoder that decodes the input sequence into a more readable and desirable output format. It is also dependent upon the time stamp. Finally, we produce a summary using the model which predicts the next relevant word by considering the previous sequence of words. This model is an extension to the feedforward network with at least one feedback connection. In standard LSTM sequence of fixed length is passed as an input to be encoded into a fixed dimension vector (v), which is then decoded into the output sequence of words. Gated Recurrent Neural Network (GRU) or Long Short Term Memory(LSTM) are used as the components of encoder and decoder since they can capture long term dependencies as shown in figure 13.

Given a dataset containing a sequence of sentences, the decoder is expected to generate the previous and next sentences, word by word. The encoder-decoder network is trained to minimize the sentence reconstruction loss, and in doing so, the encoder learns to generate vector representations which encode enough information for the decoder, so that it can generate neighboring sentences. These learned representations are such that embeddings of semantically similar sentences are closer to each other in vector space, and therefore are suitable for clustering. The sentences in our emails are given as input to the encoder network to obtain the desired vector representations.

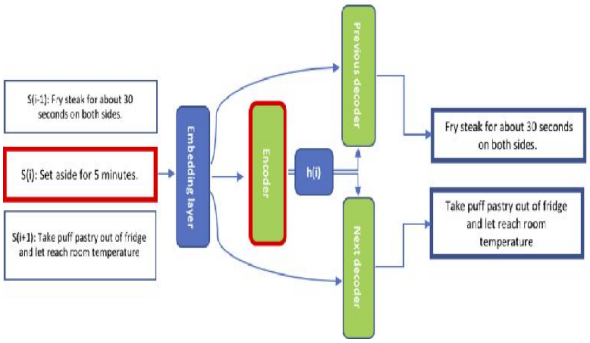
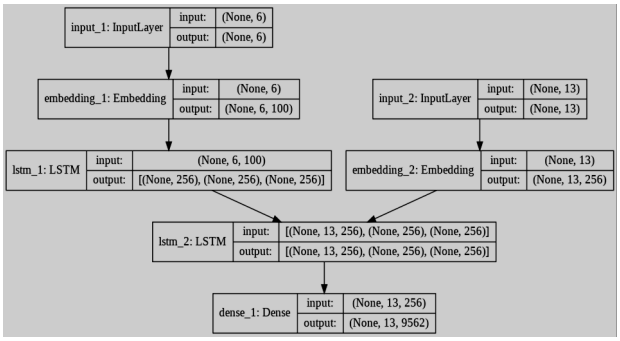


Figure 13: LSTM Model

The proposed LSTM model source code is depicted in figure 14 with the word-formation. The output is the embedding for every sentence given from the source document.




```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 1376, 150)         4500000
_________________________________________________________________
lstm_1 (LSTM)                (None, 200)               280800
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 402
=================================================================
Total params: 4,781,202
Trainable params: 4,781,202
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 14: Source Code and output for embedding

Step 7: Clustering

The clustering technique is used to cluster the embedding in high dimensional vector space. The number of clusters is equal to the desired number of sentences in the summary. It is done by using the formula the numbers of sentences in the summary to be equal to the square root of the total number of the sentence in the document.

Step 8: LSTM in summarization

The cluster of embedding sentences is interpreted as semantically similar whose meanings are expressed as one candidate sentence in the summary. The candidate is chosen based vector representation is closest to a central cluster. Candidate sentences corresponding to each cluster are then ordered to form a summary. The order of the candidate sentences in the summary is determined by the positions of the sentences in their corresponding clusters in the source document.The results are shown in figure 15,16 and 17 respectively.

```
https://en.wikipedia.org/wiki/Lists_of_diseases
valid
Web site exists
https://en.wikipedia.org/wiki/List_of_diseases_(0%E2%80%939)
valid
Web site exists
https://en.wikipedia.org/wiki/List_of_diseases_(A)
valid
Web site exists
https://en.wikipedia.org/wiki/List_of_autoimmune_diseases
valid
```

```
ence_scores.get)

summary = ' '.join(summary_sentences)

print(summary)
```

```
We developed a deep learning approach for automatically determ
ining the estimated glomerular filtration rate (eGFR) and CKD
status. A Pearson correlation coefficient of 0.741 indicated t
he strong relationship between artificial intelligence (AI)- a
nd creatinine-based GFR estimations. AI-GFR estimation offers
the possibility of noninvasive assessment of kidney function,
a key goal of AI-powered functional automation in clinical pra
ctice.
```

Figure 15: Summary of the articles

```
In [7]: # Find the element on the webpage
main_content = soup.find('div', attrs = {'class': 'syndicate','id':'topic-summary'})

In [9]: print(main_content.text)

A kidney stone is a solid piece of material that forms in the kidney from substances in the urine. It may be as small as a grai
n of sand or as large as a pearl. Most kidney stones pass out of the body without help from a doctor. But sometimes a stone wil
l not go away. It may get stuck in the urinary tract, block the flow of urine and cause great pain.
The following may be signs of kidney stones that need a doctor's help:

    Extreme pain in your back or side that will not go away
    Blood in your urine
    Fever and chills
    Vomiting
    Urine that smells bad or looks cloudy
    A burning feeling when you urinate

Your doctor will diagnose a kidney stone with urine, blood, and imaging tests.
If you have a stone that won't pass on its own, you may need treatment. It can be done with shock waves; with a scope inserted
through the tube that carries urine out of the body, called the urethra; or with surgery.
NIH: National Institute of Diabetes and Digestive and Kidney Diseases
```

Figure 16: Summary of symptoms from the Medical Wikipedia article on Cancer


```
Keywords:
kidney 0.372
ultrasound 0.149
model 0.149
kidney function 0.149
function 0.149
ckd 0.149
status 0.111
kidney ultrasound 0.111
ckd status 0.111
ai 0.111
```

Figure 17: Keyword Extracted represented as Word Cloud

3.PERFORMANCE ANALYSIS

Josef and Karel (2009) approach of evaluating the quality of the summary obtained using Precision, Recall and F-Score value. Precision(P) is the number of sentences occurring in both systems and ideal summaries divided by the number of sentences in the system summary. Recall(R) is the number of sentences occurring in both systems and ideal summaries divided by the number of sentences in the ideal summary. F-score is a composite measure that combines precision and recall. The basic way how to compute the F-score is to count a harmonic average of precision and recall:

$F=2.P.R/(P+R)$

$F\text{-Score}=(\beta 2+1).P.R/\ \beta 2.P+R$

Where β is a weighting factor that favors precision when $\beta > 1$ and recall when$\beta < 1$[3].

Table 1: Performance Evaluation

Document Size	P	R	F-Score
50 MB -100 MB	0.98	0.93	0.95
20 MB – 50 MB	0.98	0.93	0.95
5 MB – 20 MB	0.98	0.93	0.95
< 5 MB	0.99	0.97	0.95

Table 1 indicates the proposed method produces a very high sensitivity and F-Score value in the automatic summarization of multiple book contents on related topics.

Table 2: Comparative Analysis

Algorithms	P	R	F-Score
Thomas et. al. (2016)	0.61	0.57	0.59
Bhaskar et. al. (2012)	0.52	0.17	0.27
Proposed Methods	0.99	0.97	0.95

Table 2 clearly indicates that the proposed method provides a very high F-Score, Precision and Recall compared to the other similar works based on the literature review.

4. CONCLUSION & FUTURE ENHANCEMENT

The proposed framework is used to extract quality summary and keyword from multiple sources. The quality is evaluated by using the precision and recall technique. The precision and recall value for the proposed architecture is more than 95%. The same architecture can be extended in the future to summarize low resourceful languages like Telugu, Hindi, and Tamil, etc., It can further be extended to perform multimedia summarization.

REFERENCES

1. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, SharanNarang, Michael Matena,YanqiZhou,Wei Li and Peter J. Liu (2019), Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, arXiv preprintarXiv:1910.10683, 2019.
2. D. Kosmajac and V. Kešelj, "Automatic Text Summarization of News Articles in Serbian Language," 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 2019, pp. 1-6. https://doi.org/10.1109/INFOTEH.2019.8717655
3. D.D.A. Bui, G. Del Fiol, S. Jonnalagadda, PDF text classification to leverage information extraction from publication reports. J Biomed. Inf. 61, 141–148 (2016) https://doi.org/10.1016/j.jbi.2016.03.026
4. D.D.A. Bui, G. Del Fiol, S. Jonnalagadda, PDF text classification to leverage information extraction from publication reports. J Biomed. Inf. 61, 141–148 (2016) https://doi.org/10.1016/j.jbi.2016.03.026
5. Jain, Amita and Vij, Sonakshi and Tayal, Devendra K., Text Summarization Using WordNet Graph-Based Sentence Ranking, Proceedings of 2nd International Conference on Communication, Computing and Networking, Springer Singapore,711-715(2019)
6. Ibtihal S. Makki, Fahad Alqurashi: An Adaptive Model for Knowledge Mining in Databases “EMO_MINE” for Tweets Emotions Classification pp. 52-60, 7(3) (2018) https://doi.org/10.30534/ijatcse/2018/04732018
7. JR Thomas, SK Bhartiand, KS Babu, Automatic Keyword Extraction for Text Summarization in e-Newspapers, Proceedings of the International Conference on Informatics and Analytics, ACM, 2016, 86–93. https://doi.org/10.1145/2980258.2980442
8. M. song et al., PKDE4J: Entity and relation extraction for public knowledge discovery. J. Biomed. Inf. 57, 320–332 (2015)
9. MagantiSyamala andN.J.Nalini: A Deep Analysis on Aspect based Sentiment Text Classification Approaches. International Journal of Advanced Trends in Computer Science and Engineering. pp. 1795 – 1801, 8(5) (2019) https://doi.org/10.30534/ijatcse/2019/01852019
10. N. Patil, N. Patil, B.V. Pawar, Survey of named entity recognition systems with respect to Indian and foreign languages. International Journal Computer Applications 134(16) (2016) https://doi.org/10.5120/ijca2016908197
11. Priyanka Thakur and Dr. Rajiv Shrivastava: A Review on Text Based Emotion Recognition System. International Journal of Advanced Trends in Computer Science and Engineering. pp. 69-71, 7(5) (2019) https://doi.org/10.30534/ijatcse/2018/01752018
12. S.R. Kundeti, J. Vijayananda, S. Mujjiga, M. Kalyan, Clinical named entity recognition: challenges and opportunities. 2016 IEEE International Conference on Big Data (Big Data). IEEE (2016)
13. Van Lierde, H., Chow, T.W.: Incorporating word embeddings in the hierarchical Dirichlet process for query-oriented text summarization. In: 2017 IEEE 15th International Conference on Industrial Informatics (INDIN). pp. 1037–1042. IEEE(2017)
14. Wang, W.M., See-To, E.W.K., Lin, H.T., Li, Z.: Comparison of automatic extraction of research highlights and abstracts of journal articles. In: Proceedings of the2nd International Conference on Computer Science and Application Engineering. pg. 132. ACM (2018) https://doi.org/10.1145/3207677.3277979
15. Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning-based natural language processing. IEEE Computer Intell. Mag. 13(3), 55–75 (2018)
16. Zhang, Z., Petrak, J., Maynard, D.: Adapted text rank for term extraction: a generic method of improving automatic term extraction algorithms. Procedia Computer Science 137, 102–108 (2018) https://doi.org/10.1016/j.procs.2018.09.010