International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 05 | May 2023 www.irjet.net p-ISSN: 2395-0072

QUICK GLANCE: UNSUPERVISED EXTRACTIVE SUMMARIZATION MODEL

L.Sumathi [1], A.Selvapriya [2]

[1] *Assistant Professor, Government College of Technology, TamilNadu, India,*
[2]*PG Scholar, Government College of Technology, TamilNadu, India.*

Abstract - *It is an era of fast movement, where people wants to know the essence of any document (technical or non-technical) at the first glance in a fast manner. This motivates for the proposed GUI based unsupervised extractive summarization model Quick Glance, which summarizes the user input. The proposed approach uses BERT based pre-trained sentence embedding vectors and attention model to derive attention score for each sentence present in the user input. Next, Principal component analysis is applied on the attention scores to automatically to evaluate the importance of each sentence and to determine the number of sentences to be extracted Experimental results show that the proposed model is user friendly and exhibits acceptable performance and is highly recommendable for non – critical application like review, news glance and so on.*

***Key Words*: Text Summarizer, BERT, PCA ,Extractive Summarization, Linear Programming**

1. INTRODUCTION

In the modern era, where a vast amount of information is accessible online, it is crucial to offer an enhanced system for swiftly and effectively extracting the information. It is quite challenging for humans to manually extract the summary from a lengthy written document. Finding appropriate documents from the many documents accessible and learning essential information from them presents a challenge. Automatic text summary is crucial for resolving the aforementioned two issues.

The technique of extracting the most significant information from a document or group of related texts and condensing it into a concise version while maintaining its overall meanings is known as text summarizing. a briefer version of the information in the original text. Automatic text summary aims to deliver the source text in a concise, semantically-rich form. The main benefit of adopting a summary is that it shortens the reading process.

Choosing to read an article mainly depends on the size of it and the time to be spent on reading it. If the article contains less critical information and contains large amount of textual data, people tend to skip it due to its less important information and large amount of time required to consume it. Here, the articles with less critical information contain repetitive contents, which could be shrunken. But it requires deep understanding of the semantics present in the document in order to extract the most informative piece of data which can reduce the time that being spent on reading an article. So that many different information can be acknowledged on the time that being spent on reading a single article.

2. LITERATURE SURVEY

McDonald *et.al* [2] proposed the first ILP method for extractive summarization. It generates summaries by maximizing relevance (i.e., importance) of the selected sentences and minimizing their redundancy (i.e., similarity).McDonald *et.al* [2] represented each sentence as a bag-of-words vector with TF-IDF values. The importance scores are computed by using the positional information of the sentences and the similarity between each sentence vector and the document vector. The cosine similarity is used to compute the similarity between sentence vectors.

Berg-Kirkpatrick*et.al*[3]constructed an ILP summarization model based on the notion of *concept*, which is a set of bi-grams. The distinctive characteristic of this model is that it extracts, and compresses sentences simultaneously. The model not only selects bi-grams with high importance but also chooses whether to cut (delete) individual subtrees from each sentence's parsing tree. The objective function of this model is the following:

$$\max_{b,c} \sum_{i=1}^{|B|} w_i \cdot b_i + \sum_{i=1}^{|C|} u_i \cdot c_i,$$

where b_i and $c_i$are binary variables that indicate the selection of the *ith*bi-gram as a summary and its deletion from the parsing tree. $w_i$and $u_i$indicate the weights of bi-grams and possible subtree cuts, respectively. Additionally, the model has a constraint of maximum allowed summary length, which is determined by the user. The weights are estimated by soft margin support vector machine optimization with bi-gram recall loss function. Therefore, the model is trained in a supervised manner, which requires gold-standard summaries.

Galanis [4] also presented a supervised extractive summarization model that extracts sentences and concepts by maximizing sentence importance and diversity (i.e., minimizing redundancy). To represent sentences in a structured form, they leveraged various features, such as sentence position, named entities, word overlap, content word frequency, and document frequency. The model has a

constraint of user-defined maximum summary length. Furthermore, support vector regression (SVR) was used to estimate sentence importance.

Boudin[5]proposed a purely concept-based extractive summarization model. The objective function of this model is the following:

$$\max_i \sum_i w_i \cdot c_i + \mu \sum_k f_k \cdot t_k,$$

where $w_i$is the weight of a concept, $f_k$is the frequency of the non-stop word k in the document set, $c_i$is a binary variable indicating whether the concept $i$is selected, and $t_k$is a binary variable indicating the presence of the non-stop word k in the summary. This variable was introduced because the frequency of a non-stop word is a good predictor of a word appearance in a human-generated summary. To obtain the concept weight, heuristic counting (such as the document frequency of each concept) was used, or the supervised model was trained. The model also has a user-determined maximum summary length. It differs from others by applied sentence pruning. Sentences with fewer than 10 words were removed to improve computational efficiency.

Liu[6] proposed a simple weighting-based unsupervised extractive summarization method. For the *i-th*sentence in a document (Si), they calculated sentence scores by leveraging term frequency, similarity, positional information, and sentence length. In particular, term frequency, position score, and sentence length score are calculated as follows:

$$TF_i = \sum_{w \in sen_i} tf(w),$$

$$Position_i = \frac{n - p_i + 1}{n},$$

$$Length_i = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x_i-\mu)^2}{2\sigma^2}},$$

where $tf(w)$ denotes the term frequency of the word w, n represents the total number of sentences, pi refers to the position of S_i, $x_i$is the length of S_i, and μ and σ denote the mean and standard deviation of the total sentence length, respectively. For a given sentence vector Es and title vector Et, similarity score is defined as the cosine similarity of two vectors:

$$Similarity_i = \frac{\vec{s} \cdot \vec{t}}{|\vec{s}||\vec{t}|}.$$

The vectors of each sentence and article title are built based on the term frequency weighting schema. Therefore, the final sentence score is calculated as follows:

$$Senscore_i = \lambda_1 TF_i + \lambda_2 Position_i + \lambda_3 Similarity_i + \lambda_4 Length_i,$$

where λ1, λ2, λ3, and λ4 are user-defined hyper-parameters. Liu et.al [6] generated a summary by selecting sentences with the highest score until the total summary did not exceed the user-defined summary length.

3. PROPOSED MODEL

Proposed Quick Glance architecture depicted in figure 1 uses Jang et.al [1] two-step procedure for text summarization .It involves document representation and representative sentence selection. Document can be represented as a continuous sentence vector, with the usage of pre-trained Bidirectional Encoding Representations from Transformers (BERT) [8] model . Importance of the sentence is determined using ILP and PCA to pick representative sentences for the summary.

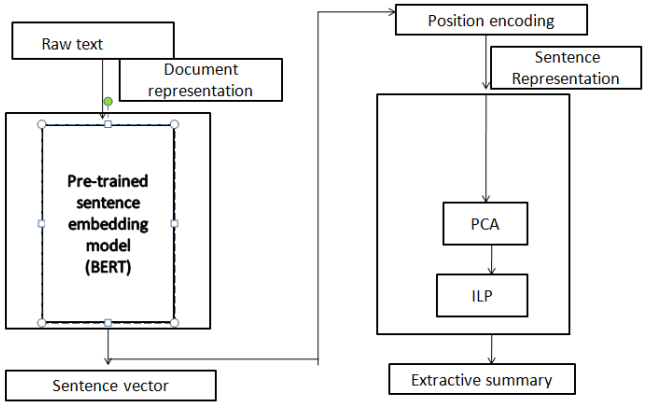

Figure 1 : Proposed System Architecture

The input document, undergoes a pre-processing step which involves tokenizing the white space, a lower-case conversion, removing the punctuations, numbers and empty strings from each token.

Since, the document consists of n sentences having the sequential information, representing the document as a sequence of sentences as follows:

D = [s1,s2, . . . ,sn]

where D denotes the document and s_k refers to its k-th sentence. Let a sentence be represented as a column vector. Thus, D_{basic}, the basic representation of D, becomes the following matrix:

D_{basic} = [sv1,sv2, . . . ,svn]

where sv_k denotes the pre-trained sentence vector of s_k .D_{basic} is a d × n matrix, where d is the embedding dimension. It is possible to use any sort of sentence embedding method

to generate sv_k. Here, the pre-trained sentence embedding model BERT (Bidirectional Encoding Representations from Transformers) as proposed by Devlin et.al [8] is used, since it has a good memory to know the positional sequence. It has pre-defined vectors for each word in a sentence. It also has the vector for sub-word if the word doesn't exists. BERT generates contextualized represents of sentence as a $T \times d_h$ matrix, where T is the number of tokens and d_h is the dimension of the hidden states. As the representation of a sentence should be a vector, the average of the BERT representations is considered to obtain the dimension d_h.

Although the matrix D_{basic}contains the intrinsic meaning of each sentence, it lacks positional information, which plays a critical role in natural language tasks. Therefore, positional encoding is used to effectively reflect sequential information, which employs cosine and sine functions. In particular, the positional encoding matrix PE is calculated as follows:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d})$$

where pos is the position, and i is the dimension. Then, the final input embedding matrix D_{emb} is calculated as the addition of D_{basic} and PE.

$$D_{emb} = D_{basic} + PE$$

To obtain a deeper representation of the document, scaled dot product attention as proposed by Vaswani et.al [7] is used , wherein the attention weights can be calculated without training parameters. The matrix **Q** is regarded as a set of queries. Likewise, the matrices **K** and **V** are the sets of keys and values, respectively. Then, the result of the scaled dot product attention is as follows:

$$Attention(Q, K, V) = softmax(Q.K^T / \sqrt{d_k})V$$

where d_k is the dimension of queries and keys.

Deep representation D_{deep} of the document is calculated as

$$D_{deep} = Attention(D_{emb}, D_{emb}, D_{emb})$$

where D_{deep} has dimension d ×n, the same dimension as that of D_{emb}. Therefore, the matrix D_{deep} can be considered the sequence of the deep representation vectors of the sentences constituting the document.

$$D_{deep} = [sv_{deep,1}, sv_{deep,2}, \ldots, sv_{deep,n}]$$

The proposed QUICK GLANCE automatically determines the number of appropriate summary sentences for each document by PCA, and thus the importance of each sentence can be quantitatively evaluated. PCA is a method for reducing high-dimensional data to lower dimensions. The principal components (PCs) extracted by PCA are composed of a linear combination of the original variables. Hence, PCA has been widely used for dimensionality reduction because it is possible to explain the entire dataset through a few PCs. The purpose of PCA is to maximize the variance of y = Xw, which is the projection of the original data X, where X is an n × p matrix and w is a vector of size p × 1. n is the number of observations and p is the number of variables. For a centered dataset X, PCA is performed by the following optimization:

$$Max\ Var(Y) = w^T \sum w$$

$$s.t\ ||w|| = 1$$

where $\sum$ is the covariance matrix of X. Solving the above equation for w yields the eigenvectors e of the covariance matrix. Therefore, when the i-th largest eigenvalue λ_i and its corresponding eigenvector e_i are given, the i-th PC is calculated as follows:

$$y_i = e_{i1}X_1 + e_{i2}X_2 + \ldots + e_{ip}X_p$$

Furthermore, by the following equation, the variance of y_i is λ_i.

$$Var(y_i) = e^T_i \sum e_i = \lambda_i$$

As the total population variance is $\sum^p_{i=1}\lambda_i$, the variance preservation ratio of y_i is

$$\alpha_i = \lambda_i / \sum^p_{i=1}\lambda_i$$

where α_i is the variance preservation ratio of Y_i . This implies that a PC y_i preserves $(100 \times \alpha_i)\%$ of the total variance. As D_{deep} has sentence vectors as column vectors, each sentence vector is considered as a variable and the dimension of the vector, d, as the number of observations. Therefore, PCA reduces the number of sentences in a document by extracting the PCs. Each PC(i.e.,) principal sentence, could be considered a vector of condensed sentences that contains as much information of the original document as possible. Subsequently, a user-defined specific hyperparameter β - the variance preservation ratio that automatically decide the appropriate number of summary sentences N. Therefore, it could be considered that at least β% of the total information is preserved in N selected sentences. Then the sentence importance score is calculated based on the correlation between the sentence vectors of D_{deep} and the PS's. As a PS is a high-level vector that efficiently condenses the intrinsic information of the document, greater similarity of a sentence vector to a PS implies higher importance of the sentence. Therefore, sentence importance score is defined by:

$$imp(s_i) = \sum_{k \neq i} \cos(sv_{deep,i}, PS_k)$$

where $imp(s_i)$ is the sentence importance score of sentence $s_i$and cos denotes cosine similarity.

The optimization problem is formulated as in McDonald et.al [2], because the minimum extraction unit is a sentence, not a concept. The requirements for the formulation are sentence importance scores and similarity scores between sentences. The similarity between the sentences s_i and s_j is defined as the cosine similarity of their deep sentence representation vectors as follows:

$$sim(s_i,s_j) = cos(sv_{deep,i},sv_{deep,j})$$

Then, the appropriate number of summary sentences (the length of summary sentences) for each document is automatically determined through PCA as follows:

$$\sum^{n}_{i=1} x_i = N$$

where N is the appropriate number of summary sentences.

In order to reduce the time complexity of QUICK-GLANCE from $O(n^2)$, sentence pruning is performed to remove unimportant sentences. The sentences to be extracted are those which should have high sentence importance scores and low redundancy scores. Therefore, the pruning score of a sentence s_i is defined by

$$Pr - score(s_i) = \frac{imp(s_i)}{\frac{1}{n-1}\sum_{j \neq i} sim(s_i, s_j)}$$

where $imp(s_i)$ is the sentence importance score of sentence s_i, $sim(s_i,s_j)$ is the similarity between the sentences s_i,s_j.

4. RESULTS AND DISCUSSIONS

Quick Glance Text summarization application provides Graphical User Interface on the Web using Django Web Framework. Input from user can be sent to the application as shown in Figure 2 in 3 different ways .

- Direct text using Textbox
- Parsed text from any given URL
- Textual data from a .txt file

The data from the textbox and .txt file can be retrieved using simple forms and read() method from Django. Whereas the parsed text from the URL has to be retrieved using Web scraping through BeautifulSoup.

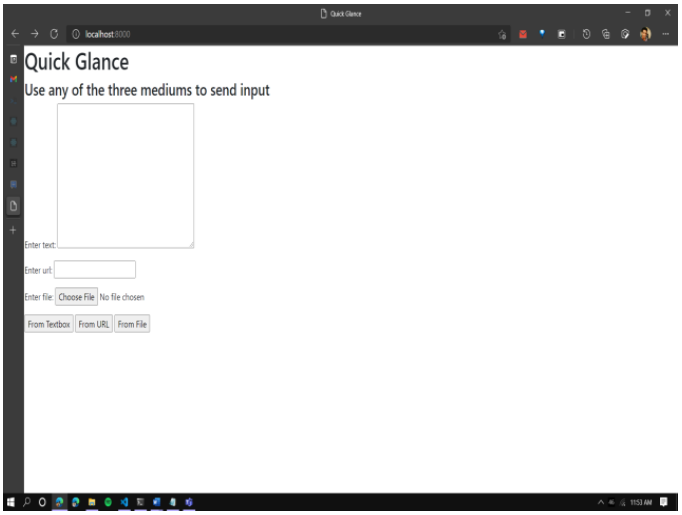

Figure 2 : GUI Homepage

The given text will be paragraphs by default, so it has to be converted into sentences for text extraction. NLTK is used for converting the given text into sentences. Using a split() function is a good idea, but with a huge amount of text which may include several quotes and special characters, it's will not always provide a good result. So, nltk.load() has been used.

After the text has been converted to sentences, it must be cleaned. It includes removing the punctuation, removing the needless spaces, converting all text to lower case, removing the number tokens. The processed text will be forwarded to further steps. The cleaned text is shown in Figure 3.

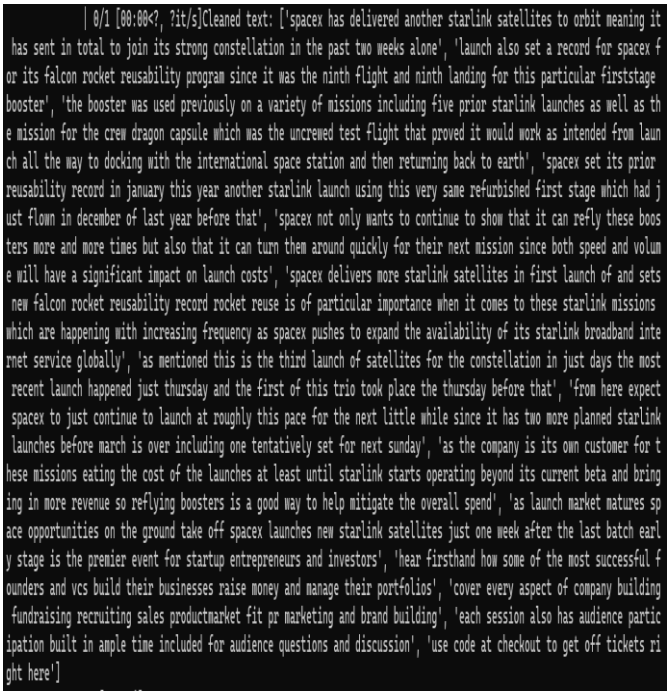

Figure 3 Cleaned Text

Positional encoding is used to effectively reflect sequential information. The positional encoding method employs cosine and sine functions to calculate the position of the word in the sentence. Result of positional encoding of given text is shown in Figure 4.

```
Positional Encoding: [[ 0.00000000e+00  1.00000000e+00  0.00000000e+00 ...  1.00000000e+00
   0.00000000e+00  1.00000000e+00]
 [ 8.41470985e-01  5.40302306e-01  8.28430762e-01 ...  9.99999994e-01
   1.02427522e-04  9.99999995e-01]
 [ 9.09297427e-01 -4.16146837e-01  9.27994032e-01 ...  9.99999978e-01
   2.04855043e-04  9.99999979e-01]
 ...
 [-9.99990207e-01  4.42569799e-03 -9.67341353e-01 ...  9.99999334e-01
   1.12670251e-03  9.99999365e-01]
 [-5.36572918e-01  8.43853959e-01 -7.51787991e-01 ...  9.99999208e-01
   1.22912996e-03  9.99999245e-01]
 [ 4.20167037e-01  9.07446781e-01  1.25201247e-01 ...  9.99999070e-01
   1.33155739e-03  9.99999113e-01]]
```

Figure 4 :Positional encoding

Attention can attend a semantically important concept or region of interest and to find the relative strength of attention paid on multiple concepts. It can switch attention among concepts dynamically according to task status and improves the focus on particular sentence. The calculated attention score for the given input is shown in Figure 5

```
Attention: [[-0.01296376  0.75655575  0.80257756 ...  0.91045055  0.38104523
   0.99572824]
 [ 0.36434821  0.4387853   1.15003223 ...  0.93387324  0.30177774
   1.16595048]
 [ 0.23134379 -0.17009876  1.01460456 ...  1.01161774  0.29661554
   1.14027814]
 ...
 [-0.72289044  0.69350871 -0.84313947 ...  0.73094768  0.04381297
   1.3410702 ]
 [-0.59430045  0.79377577 -0.32485462 ...  0.52733183 -0.04414779
   0.72696647]
 [ 0.3729566   0.96766086  0.30165624 ...  0.78898542  0.02480833
   1.353083  ]]
```

Figure 5 :Attention Scores

Method used for dimensionality reduction because it is possible to explain the entire dataset through a few PC's (Principal Components) and the same has been shown in Figure 6

```
PCA: [[-0.35303293 -0.3534824  -0.36433566 -0.38934905 -0.40098216 -0.39970898
  -0.38148502]
 [ 0.48749304  0.39471812  0.26645061  0.07986354 -0.17529672 -0.32751779
  -0.62544159]
 [ 0.45685729  0.18959625 -0.13821236 -0.38464857 -0.42672979 -0.21502675
   0.59995223]
 [-0.43873679  0.08549422  0.43423821  0.45238287 -0.34816919 -0.45027879
   0.28812307]
 [-0.00967492 -0.04088359  0.08678599  0.13086214 -0.70595691  0.67585819
  -0.1357158 ]
 [ 0.33106422 -0.27566287 -0.56387552  0.68303246 -0.08067072 -0.14961354
   0.03202409]
 [-0.35610758  0.77351814 -0.51294319  0.07593485 -0.01677336  0.06934579
  -0.02983428]]
```

Figure 6 : Principal components

In ILP, sentence scores and similarity scores between sentences are calculated whereas sentence pruning helps to reduce unimportant sentences and time complexity

The summary of the given document is achieved and is displayed in the GUI's homepage as shown in Figure 7

Figure 7 : Output after summarization

5. CONCLUSION AND FUTURE WORK

A user-friendly text summarizer, the QUICK GLANCE is proposed efficiently by retaining the intrinsic information. Experimental results revised that the Quick Glance provides well organized summary with needed information for non-critical applications like review, news glance etc., in a user-friendly manner.

Here, the appropriate number of sentences in the summary will be decided by PCA using a variance preservation ratio β. In future, PCA can be replaced with Linear Discriminant Analysis (LDA), and non-negative matrix factorization (NMF). As of now, the summarized output is displayed in the GUI's Homepage. Future works can include that the output can also be sent to the user's valid mail. It can be also implemented as a mobile application which will help the students for quick grasping of the context.

REFERENCES

[1] Jang, Myeongjun, and Pilsung Kang. "Learning-Free Unsupervised Extractive Summarization Model." IEEE Access 9 (2021): 14358-14368..

[2] McDonald, Ryan. "A study of global inference algorithms in multi-document summarization." European Conference on Information Retrieval. Springer, Berlin, Heidelberg, 2007..

[3] Berg-Kirkpatrick, Taylor, Dan Gillick, and Dan Klein. "Jointly learning to extract and compress." Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. 2011.

[4] Galanis, Dimitrios, GerasimosLampouras, and Ion Androutsopoulos. "Extractive multi-document summarization with integer linear programming and support vector regression." Proceedings of COLING 2012.

[5] Boudin, Florian, Hugo Mougard, and Benoit Favre. "Concept-based summarization using integer linear programming: From concept pruning to multiple optimal solutions." Conference on Empirical Methods in Natural Language Processing (EMNLP) 2015.

[6] Liu, Maofu, Limin Wang, and LiqiangNie. "Weibo-oriented chinese news summarization via multi-feature combination." Natural Language Processing and Chinese Computing. Springer, Cham, 2015. 581-589.

[7] Vaswani, Ashish, et al. "Attention is all you need." arXiv preprint arXiv:1706.03762 (2017).

[8] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).