Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 3 Issue 8, August - 2016

Development Of Automatic Text Summarizer For PDF Files

OYINLOYE, Oghenerukevwe Elohor[1]
Department of Computer Science
Ekiti State University, Ado-Ekiti,
Ekiti State, Nigeria
rukkivie@yahoo.com

ODUNTAN Akorede Aliu[3]
Department of Computer Science
Ekiti State University, Ado-Ekiti,
Ekiti State, Nigeria
oduntanakoredealiu@yahoo.com

OGUNDELE Fatai[2]
Department of Computer Science
Ekiti State University, Ado-Ekiti,
Ekiti State, Nigeria
ogundele.fatai.k@gmail.com

***Abstract*—FOATS is an automatic text summarizer developed to summarize text in Portable Document Format (PDF) files. As information continues to grow in digital system, many people rely so much on information saved in these formats. This system provides an opportunity to prevent wasting of time and energy in reading irrelevant materials. It can be used to determine the major decision of a text, which will guide the reader through what to read and to know whether the material is relevant or not. However, the method used in this system is divided into three major phases which include Upload sheet, Pre-processor and Summarizer. Sentence interception is used by the summarizer to generate the summary of the text. This method performs its function by determining how related sentence in each paragraph is. The system finds their intercept and uses the most important sentences as the summary of the text. FOATS was tested using five (5) different documents and a computational result was reported for the various experiments. The system performance is extremely efficient by generating the summary produced into sequence of paragraphs while retaining the major points of the text. The system is said to be 90% efficient.**

Keywords—Text Summarizer; PDF Files; Upload Sheet; Sentence Interception; Pre-Processor

I. INTRODUCTION

Text summarization is the process of reducing the length of a text by condensing the text into a shorter version while preserving the most important contents to guide and inform the reader about the text information. This information is very vital that people cannot live without it. With it, laws and customs of the society are prevented, students rely on it for their research and the entire community depends on it to survive. Because of these, people continue to face problem by wasting lots of time and energy on materials in order to derive information. Therefore to

address this problem of time wasting, summarization is always put into practice. This summarization is divided into two parts which are manual summarization and automatic summarization.

Manual summarization has been in existence over the years. It is the process by which summary is created by human through reading and understanding of a text to find the key points which are then used to generate a new sentences. This kind of summarization is considered to have two parts which are abstract that always come before the entire text and summary which comes at the end of the text. These two terms are considered to be related but distinct. According to [1] a summary is a “brief restatement within the document (usually at the end) of its salient findings and conclusions intended to complete the orientation of a reader who has studied the preceding text”; while an abstract, according to this same standard is a “brief and objective representation of a document or an oral presentation”. Despite their differences, they both serve as a summary to document.

Conversely, an automatic text summarization is the process by which a computer program generates the summary of a text by reducing the length of the text while retaining the most important fact of the text. There are two major approaches to automatic text summarizer which are extractive based method and abstractive based method. Extractive based method is the process by which the major sentence of a text is extracted out as a summary. While abstractive based method is the process by which the summary generated is represented by new sentences different from the original text yet with the same meaning so that the result aside providing adequate points, it generates concise, coherent and no redundant result.

An automatic text summarizer has provide great benefits to modern technologies by rendering its use in some services such as search engine hit, advert system, online summarizer and news summary. Despite of all this benefits, it is very essential to provide automatic text summarizer for PDF files as people rely on this format to derive information. For this reason, this research study provides an offline automatic text summarizer for PDF files. It generates its summary based on each paragraph segment using sentence interception method such that the end summary follows a coherent pattern.

II. RELATED WORK

Automatic text summarizer has being in existence over the years. The pioneer [2]in 1958 develops an automatic text summarizer that uses word significant method. In this method, a list of English common words such as 'a', 'is', 'and' and so on are removed from original text by comparing the original text with a list of common words or by using statistical techniques while regarding all other words as significant words. This set of significant words is used in scoring the sentences so that the sentences that have the highest significant words are ranked as the summary of the sentences.

Meanwhile, [3] in 1969 stated that the extracts produced by [2] were of sufficient quality to encourage further research. In contribution to this, he emphasized that "a purely statistical method of producing extracts was suspected of being inadequate". He therefore designed an indicative text summarizer which allows a user to screen a body of literature to decide which documents deserve more detail attention. This summarizer was designed using three additional methods which are pragmatic words (cue words), title and heading words and structural indicators (sentence location).

Aside from this early generation of automatic text summarizer, several other methods had been used to enhance text summarizer in order to improve its efficiency.

[4] Develop an automatic text summarizer called SUMMARIST. This summarizer is divided into four major parts which works together to produce abstractive related summary. This part includes *Pre-Processor* which deals with splitting and scoring of words ina passage. *Topic identification*, this method rank sentences based on previous method such as keywords, title words, position and cue words with others in order to select best sentences from the text after which the *topic interpretation* interprets the words in the new sentences so as to generate new set of words. In this aspect, words like cashew, mango, orange can be referred to as fruits. With this, the summarizer produces human related summary which are likely not in original text. The last part of which is *summary generator* generates the overall summary of the text.

In terms of application, automatic text summarizer had been applied to varieties of field in this modern information system part of which include: search engine hit, news summary, advert system and online text summarizer.

[5] Developed an automatic text summarizer for contextual advertising system. This system suggests suitable adverts to user while surfing the internet. The method used in this summarizer include; Pre-Processor, text summarizer, classification and matching. In the classification aspect, both page excerpts and advertisings are classified according to a given set of categories which are then used in conjunction with the original words by matcher to suggest adverts to the web page according to similarity score.

Moreover, [6] developed a web-based automatic text summarizer called FarsiSum which summarize Persian newspaper text (html) in unicode format. It uses modules implemented in [7] with exception of lexicon which was replaced by stop-list to generate summary.

Meanwhile [8] developed a graph-based abstractive summarizer for text with redundant opinions. This kind of text is mostly found in users comment. According to him, "certain text cannot be properly summarized by extractive method especially those with redundant opinions, no matter which single sentence is chosen as a summary, the generated summary will be biased". For this reason, abstractive related summarizer was developed to generate a more coherence summary of text in more informative way.

[9] Stated more other methods which had been applied to abstractive text summarizer.

III. DESIGN OF THE SYSTEM

FOATS is designed as shown in figure 1. This is an offline standalone application which takes in input text, PDF files or text document and perform a summarization on it using three phases; Upload Sheet, Pre-Processor and Summarizer. The system will check for proper input using the upload sheet and send the result to processor. The pre-processor will extract sequence of paragraph from the text and send its output to processor to perform further process. The processor summarizes the text using sentence interception method. However, the advantages of this summarizer is not limited to the ability to summarize PDF formats file but also the ability to summarize based on each paragraph and return an organized concise summary.

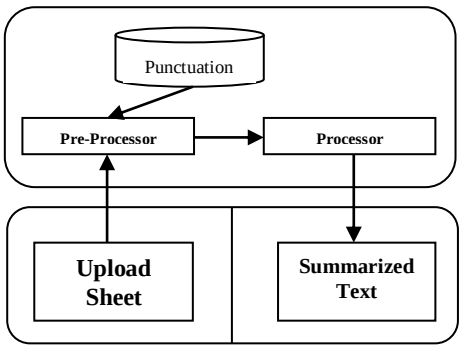


Figure 1 FOATS Architecture

IV. PHASES OF THE SYSTEM

A. Upload Sheet

This phase will interacts with the user and determine not only the text format but also the kind of text upload by the user. Immediately the system receives the text, it will process it and appropriate response will be send back to the user. The operation of this phase is achieved by checking if the text is not authorized, corrupt or the file location fails. If any of this occurs, an error message will be sent to the user. But if the text has no problem, it will open and at the same time send to the pre-processor for further operations.

B. Pre-Processor

The pre-processor will act as a scanner and read the input text received from the upload sheet. This text is segmented into various paragraphs in which will be send to the summarizer before summary can take place. The list of paragraphs such as paragraph 1 to the last paragraph will be extracted out of the text and store in a specific place for summary. For this reason, any word that exist within the paragraph will be opportune to be selected as candidate for text summary while others that fails to comply with the rules such as words that exist among figure, tables and equations will have no place in the chosen list of words since they are not part of the paragraph. This is done because from a paragraph, a topic sentence for good summary can be known. The major distinction of this phase is the ability to remove unwanted part such as comment, figure, tables, and equations and so on while retaining the headings and the list of paragraphs under each heading. This helps to follow strictly the rules of generating a good summary.

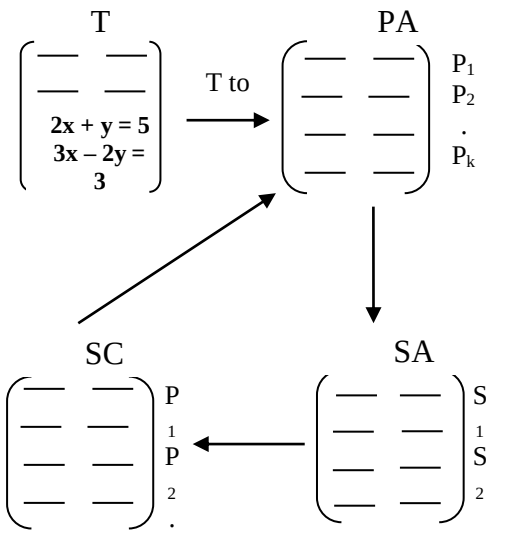
C. Summarizer (Processor)

The summarizer will receive list of words, sentences and paragraphs from the pre-processor. These lists are used by the summarizer as discussed in the mathematical model to summarize text based on each paragraph. The summary generated is organized into paragraphs which can be reorganized by the user to their satisfaction.

Moreover, the basic method of this text summarizer is sentence interception. This method relates every sentence of a paragraph together to determine how related the sentences are. The sentence that has the best relationship is used as the best sentence of the paragraph.

V. MATHEMATICAL MODEL

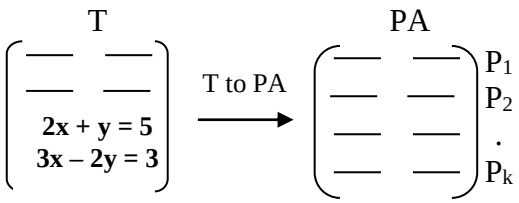
The mathematical model for the system "**FOATS"** is divided into four basic steps as shown in equation 1.



(1)

Step 1:

After the original text **T** have been uploaded to the system which contain series of paragraphs **P** with different kind of figure, tables and diagrams. The Paragraph Counter **PC** will locate and count the paragraph **P** in the original text **T** such that the paragraphs ranges from (P_1, P_2, P_3, . . .,P_k). Then, the Paragraph Extractor **PE** will identify and extract the entire paragraph **P** in the original text **T** ignoring all the figure, tables and diagrams and store the result in an Array **PA**. This is illustrated in equation 2.



(2)

- Let **PA** be an Array (ordered as a list or queue).
- Let **T** be the Original Text.
- Let **K** be an Integer Variable.
- Let **P** be a Paragraph.
- Let **PC** be a Paragraph Counter which will locate and count Paragraph **P** in the Original text **T**.

- Let **PE** be Paragraph Extractor (which will identify and extract all Paragraph **P** in Original Text **T**).

The conversion of original text **T** to plain text **PA** (sequence of paragraphs without diagrams, Figures or tables) is as follows:

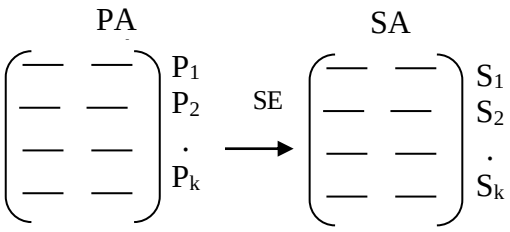
- **K** ⟵ **PC**
  The Paragraph Counter **PC** counts the number of the Paragraph **P** in the Original Text **T** and store the result in **k**.
- **PA** = [ ] / max (**PA**) = **k**

Declare PA as an array such that the maximum capacity of PA equals to integer K.

- **PA** = [**PE**] such that **PE** = [$\{\mathbf{P}\}^{k}_{n=1}$] $1 \leq \mathbf{P} \leq \mathbf{K}$
  Store the content of the paragraph extractor PE into PA such that paragraph extractor PE itself is a method consisting of other methods used to return an array storing the bounded sequence of paragraph **P**

Step 2:

After the original text **T** has been deduced to paragraph **P** ignoring all the Figure, tables and diagrams and stored in an array **PA**. The next step is to generate a summary sentence **S** for each paragraph **P**. For example "*paragraph 1*"$\mathbf{P_1}$ produces "*sentence 1*"$\mathbf{S_1}$, and so on up to the last paragraph $\mathbf{P_k}$ which will produce sentence $\mathbf{S_k}$.The summary extractor **SE** will pick paragraph **P** and return most ranked sentence **S** in each paragraph**P** as the summary of the paragraph (i.e $\{P\}^{k}_{n=1}$ = SE = $\{SE\}^{k}_{n=1}$) and store them in an array **SA**. This step is further illustrated in equation 3.



(3)

- Let **SA** be an array or variable storing list of summary sentences sequentially.
- Let **S** be a summary sentence generated from paragraph **P** such that ($\{P\}^{k}_{n=1}$ = $\{SE\}^{k}_{n=1}$) i.e paragraph 1 produces summary sentence 1.
- Let **SE** be a summary extractor such that **SE** picks a paragraph **P** and return most ranked sentences **S** of the paragraph as the summary of the paragraph.

Therefore, the conversion from **PA** to **SA** is as follows:

- **SE** ⟵ **PA**
  Copy the paragraph in PA to SE and produce the equivalent summary.
- **SA** ⟵ **SE**

Store the summary produce by SE to SA

Step 3:

After generating sentence **S** from each paragraph **P**, the next step is as follows;

Suppose **SA** stores a sequence of sentence $\{S_n\}^{k}_{n=1}$, a threshold **j**, where **j** is an integer number will be used to group $\{S_n\}^{k}_{n=1}$such that each group is a paragraph. Then, **j** will be equal to total number of sentences in each new paragraph. For example, suppose **j** = 4, then every four sentences will be grouped as a paragraph. This is illustrated in equation 4.

$$\begin{matrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{matrix} \Big\} P_1 \qquad \begin{matrix} S_5 \\ . \\ . \\ .S_n \end{matrix} \Big\} P_n \qquad \textbf{(4)}$$

Finally, this result will be concatenate to a new set of paragraph which is the initial summary.

Step 4:

If the initial summary generated is less than or equal to **M** (where **M** is the required summary length of the paragraph), the summary will be display, printed or saved as PDF file based on user's choice. Otherwise, if the initial summary generated is more than **M**, the system will go back to step 2.

VI. Summary Extractor (SE)

This is a method that picks up a paragraph, summarize it and return the summary which is a single sentence from the paragraph. This method uses sub-methods which are discussed below to achieve its tasks.

A. Awarding Score Using Sentence Interception

This is the process by which the frequency of words in a paragraph can be determined using sentence interception. In this case, given a paragraph **P**, the sentence in the paragraph will be intercepted to one another while neglecting the interception of a sentence to itself. This process is similar to Text Superstructure in English Language (sentence – to – sentence relationship).

To achieve this, suppose the given paragraph **P** contains **n** number of sentences, an iterating variable **i** and **j** will be used on the sentence such that a sequence of sentence $\mathbf{\{S_i\}^{n}_{i=1}}$ with bounded range of **[i, n]**. However, the result of this sentence interception will give a set of words that appears in both sentence **i** and sentence **j**. Meanwhile, the number of this words will be counted and recorded to give a transpose matrix with row **i** and column **j** written as **[si, sj]**. At this point, the score of each sentence can be calculated as the sum of row or column (since it gives transpose matrix) an illustration is shown below.

Let paragraph **P** has n number sentences such that:

$\{Si\}^{n}_{i=1}$ = Sequence of sentence Si from sentence 1 to last sentence
$\{S_j\}^{n}_{j=1}$ = Sequence of sentence Sj from sentence 1 to last sentence in paragraph **P**
Therefore, the interception of all sentences in paragraph P is illustrated in Table 1:

Table 1: Sentence Interception

$\{Si\}^{n}_{i=1}$ intercept $\{S_j\}^{n}_{j=1}$

[i, j]	S_1	S_2	S_3	…….	S_n
S_1	------	$[S_1, S_2]$	$[S_1, S_3]$	…….	$[S_1, S_n]$
S_2	$[S_2, S_1]$	--------	$[S_2, S_3]$	…….	$[S_2, S_n]$
S_3	$[S_3, S_1]$	$[S_3, S_2]$		…….	$[S_3, S_n]$
.	.	.		…….	
S_n	$[S_n, S_1]$	$[S_n, S_2]$	$[S_n, S_3]$	…….	--------

Therefore, the score of each sentence is attached to its paragraph as:({sentence i}$^{n}_{i=1}$, {sentence i}$^{n}_{i=1}$) such that

When i = 1 S_1 = Score 1 = $[S_1, S_2]$ + $[S_1, S_3]$ + . . . + $[S_1, S_n]$

When i = 2 S_2 = Score 2 = $[S_2, S_1]$ + $[S_2, S_3]$ + $[S_2, S_n]$ **(5)**

When i = n S_n = Score n = $[S_n, S_1]$ + $[S_2, S_3]$ + . . . + $[S_n, S_{n-1}]$

B. Ranking and getting the best Sentence

The best sentence can be achieved by sorting the scores. The sentence with the highest score will be picked as the most ranked sentences.

VII. IMPLEMENTATION AND TESTING

The system was implemented using JAVA Netbean and tested with five different documents. A computational result was obtained based on number of extracted paragraph, total words of the text and total number of summary generated as shown in Table 2

Table 2: Experimental Result

S/N	Extracted Paragraph	Total Words	Total word of Summary
1	5	280	113
2	6	653	212
3	6	770	218
4	6	269	194
5	5	594	139

VIII. SUMMARY/CONCLUSION

Automatic text summarizer have been designed for many applications such as search engine hit, advert system and news summary. Despite of this various applications, automatic text summarizer is needed to summarize text in PDF format. For this reason, FOATS is designed as an offline summarizer to summarize text in PDF files.

The summarizer uses sentence interception method to generate summary. Moreover, the summarizer is implemented using JAVA and the system is tested. The summary generated can be concatenated into sequence of paragraphs and made available for users to print.

A. Limitations

The limitation is majorly on pre-processor which is not fully completed to segment paragraphs into its various headings.

B. Recommendations

The pre-processor should be made to organize paragraph properly and arrange them under each headings so that the summary generated will be organize under each heading.

REFERENCES

[1] National Information Standards Organization (1997); Guidelines for Abstracts; Revision of ANSI Z39.14 – 1979 (R1987); ISSN: 1041 – 5653.

[2] Luhn H.P. (1958); The Automatic Creation of Literature Abstracts; IBM Journal of research and Development, Pg 159 – 165.

[3] Edmundson H.P (1969); New Method in Automatic Extracting; Journal of the Association for Computing Machinery, Vol. 16, No 2, Pg 264 – 285.

[4] Eduard Hovy and Chin-Yew Lin (1998); Automatic Text Summarization in SUMMARIST;
Information Sciences of the University of Southern California, Pg 1 – 14.

[5] GiulianoArmano, Alessandro Giulian, and EloisVargin (2010); Experimenting Text Summarization Techniques for Contextual Advertising, Pg 1 – 10.

[6] Mazdak, N. (2004). *FarsiSum - a Persian text summarizer*, Master thesis, Department of Linguistics, Stockholm University, Sweden.

[7] Dalianis, H. (2000). *SweSum - A Text Summarizer for Swedish*, Technical report, TRITA-
NAP0015, IPLab-174, NADA, KTH, October 2000.

[8] KavitaGanesan, ChengXiangZhai and Jiawei Han (2010); Opinosis: A Graph-Based Approach to Abstractive Summarization of Highly Redundant Opinions; Department of Computer Science, University of Illionis at Urbana- Champaign, Pg 1 – 3.

[9] Atif Khan and NaomieSalin (2014); A Review of Abstractive Summarization Methods; Faculty of Computing, UniversitiTecknologi Malaysia, Pg 1 – 9.