# CRESS: Efficient Long Document Summarization

Aziz Amari
Software Engineering and Mathematics Department
INSAT
Tunis, Tunisia
aziz.amari@insat.ucar.tn

***Abstract*—In the era of extensive textual information, effective summarization of long documents is paramount. This paper presents CRESS (Comprehensive Representation Extraction and Summarization System), a novel approach designed to address the challenges of summarizing lengthy content efficiently and cost-effectively. CRESS leverages clustering, mapping, and reduction techniques to distill comprehensive summaries from long documents, while optimizing both performance and cost-effectiveness. Practical applications demonstrate the potential of CRESS in condensing extensive textual content, providing a valuable tool for knowledge extraction and dissemination in various domains.**

***Index Terms*—Large Language Models, Summarization, Natural Language Processing, Context Window, Vector Databases, Vector Embeddings, GPT, Prompt Engineering**

## I. Introduction

In today's data-rich landscape, the concise summarization of lengthy documents has assumed paramount importance. Traditional summarization techniques [1] have gradually yielded the spotlight to Large Language Models (LLMs), heralding an era of potentially more effective outcomes [2]. However, the adoption of LLMs, whether through local inference or API access, has revealed a notable concern: the significant expenses incurred, particularly when tasked with summarizing long documents.

Summarizing book-length documents (>100K tokens) that exceed the context window size of large language models (LLMs) requires first breaking the input document into smaller chunks and then prompting an LLM to merge, update, and compress chunk-level summaries.

This paper introduces CRESS, the Comprehensive Representation Extraction and Summarization System. CRESS emerges as a response to the intricacies of long document summarization, conceived to deliver a cost-effective alternative.

Within the forthcoming sections, we will delve into the conventional incremental updating method [3], scrutinizing its resource-intensive and costly nature. This examination will emphasize the challenges and limitations of traditional approaches. Subsequently, we will introduce CRESS as an alternative to address these challenges.

Identify applicable funding agency here. If none, delete this.

## II. Background

### A. Large Language Models

In the realm of language modeling, the term 'large language models' (LLMs) typically refers to Transformer-based language models characterized by their vast parameter counts, often numbering in the hundreds of billions or more. These models are meticulously trained on extensive textual datasets [4], with notable examples including GPT-3, PaLM, Galactica, and LLaMA. LLMs are renowned for their remarkable proficiency in comprehending natural language and tackling intricate language-related tasks. Additionally, with the ascent of instruction-tuned LLMs like ChatGPT, their adaptability and utility continue to expand especially in the field of summarization [5].

### B. Context Windows

A context window refers to the extent of text that a model can effectively process in a single operation. For example, GPT-4's maximum context window spans 32,000 tokens, equivalent to the length of a college thesis, while 256,000 tokens are comparable to the length of a novel. This limitation becomes pivotal when summarizing lengthy documents, where content often exceeds the boundaries of the context window, potentially leading to information loss and compromising summarization quality.

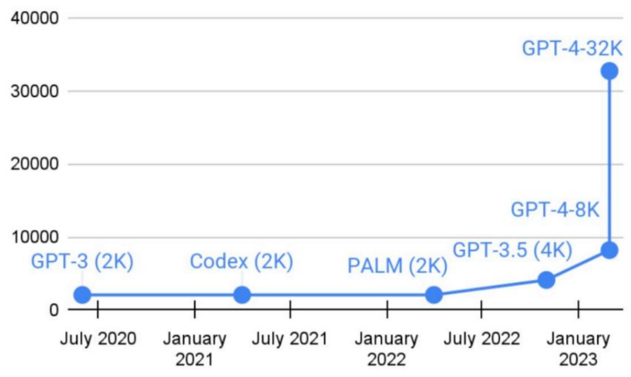

Fig. 1: The context lengths of foundation models. Source: Dan Fu (Twitter @realDanFu)

### C. *Vector Databases*

[8] It is increasingly common that rich, unstructured data such as large texts, images [7] and video are not only stored, but given semantics through a process called vectorization [6] which transforms the data into n-dimensional vectors consisting of natural, real, or complex numbers. The challenge of working with vector embeddings is that traditional scalar-based databases can't keep up with the complexity and scale of such data, making it difficult to extract insights and perform real-time analysis. That's where vector databases come into play – they are intentionally designed to handle this type of data and offer the performance, scalability, optimized storage, querying capabilities, and flexibility you need to make the most out of your data. As this is not the main focus of our paper we'll be content with just the high-level definition you can read more about the topic: [9] [10] [11]

## III. Incremental updating [3]

### A. *Definition*

Before discussing our CRESS approach we have to outline incremental updating—for prompting an LLM to summarize book-length documents that exceed its maximum context size. The length of the input document necessitates first dividing it into smaller chunks and then iterating through each chunk in order while continuously updating a global summary with more information, you could think of this as Map-Reduce.

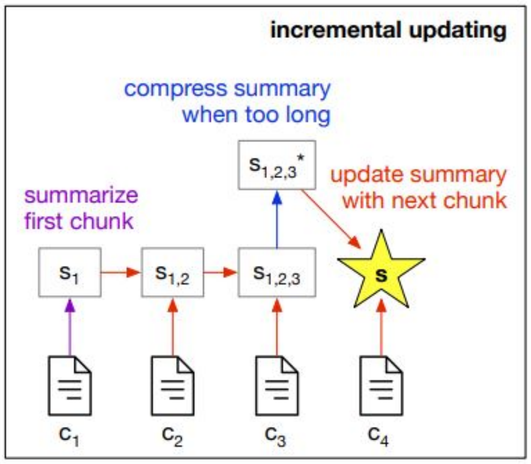

Fig. 2: A global summary is updated and compressed as we step through the book chunk-by-chunk.
Source: [3]

### B. *Example Implementation of Incremental updating*

Here is a sample implementation using prompts from [3]

- **Generate initial summary**
  "'Below is the beginning part of a story:
  INSERT BEGINNING
  We are going over segments of a story sequentially to gradually update one comprehensive summary of the entire plot. Write a summary for the excerpt provided above, make sure to include vital information related to key events, backgrounds, settings, characters, their objectives, and motivations. You must briefly introduce characters, places, and other major elements if they are being mentioned for the first time in the summary. The story may feature non-linear narratives, flashbacks, switches between alternate worlds or viewpoints, etc. Therefore, you should organize the summary so it presents a consistent and chronological narrative. Despite this step-by-step process of updating the summary, you need to create a summary that seems as though it is written in one go. The summary should roughly contain n words and could include multiple paragraphs.
  Summary:"'
- **Generate intermediate summaries**
  "'Below is a segment from a story:
  INSERT SEGMENT
  Below is a summary of the story up until this point:
  INSERT SUMMARY
  We are going over segments of a story sequentially to gradually update one comprehensive summary of the entire plot. You are required to update the summary to incorporate any new vital information in the current excerpt. This information may relate to key events, backgrounds, settings, characters, their objectives, and motivations. You must briefly introduce characters, places, and other major elements if they are being mentioned for the first time in the summary. The story may feature non-linear narratives, flashbacks, switches between alternate worlds or viewpoints, etc. Therefore, you should organize the summary so it presents a consistent and chronological narrative. Despite this step-by-step process of updating the summary, you need to create a summary that seems as though it is written in one go. The updated summary should roughly contain n words and could include multiple paragraphs.
  Updated summary:"'
- **Compression**
  "'Below is a summary of part of a story:
  INSERT SUMMARY
  Currently, this summary contains n words. Your task is to condense it to less than m words. The condensed summary should remain clear, overarching, and fluid while being brief. Whenever feasible, maintain details about key events, backgrounds, settings, characters, their objectives, and motivations - but express these elements more succinctly. Make sure to provide a brief introduction to characters, places, and other major components during their first mention in the condensed summary. Remove insignificant details that do not add much to the overall storyline. The story may feature non-linear narratives, flashbacks, switches between alternate worlds or viewpoints, etc. Therefore, you should organize the summary so it presents a consistent and chronological narrative.
  Condensed summary:"'

### C. *Where it fails*

This seems great and it functions well but the model will have to process all text as input anyway, even if not at once. Let's imagine we wanted to summarize a novel, that's almost 200k tokens, using gpt-4-32k at a rate of $0.06 per 1k tokens which will cost us around $12. You may argue that we can self-host an LLM instead of using gpt-4-32k through an API but the costs will not be that different considering you have to pay the cloud provider.
Is there a way to do this without having to go through all the text?

## IV. CRESS Approach

The Six-Step process:

### A. *Collect the Entire Text (C)*

This is the most straightforward part, simply fetch your data whether it's on your disk, a cloud virtual machine, parsing a PDF file, transcribing an audio clip,..

### B. *Resegment into Manageable Chunks*

The next step is splitting the long text into many chunks so they can fit in the context window of a Large Language Model, for example, we can choose 4k-tokens chunks. Tools and libraries like LangChain [12] offer existing methods for doing this such as ”TokenTextSplitter”.

### C. *Embed Text Chunks as Vectors (E)*

Here you would use an embedding model like OpenAI's model [10] to embed all text chunks, if the text is really long you can store the vectors in a Vector Database otherwise for most applications keeping the vectors in memory works.

### D. *Streamline Vector Clusters (S)*

This step consists of applying clustering algorithms [13] [14] on the vector representations obtained in the previous step to group our chunks into k-clusters. We can find the best value for k [15] through techniques like the Elbow-Method. The vectors are n-dimensional so we can't visualize them but it's interesting to apply a Dimensionality Reduction [16] algorithm like Principal Component Analysis [17] to downscale the vectors to 2-dimension, figure 3.

### E. *Select the Most Representative Vectors (S)*

Now how will we get the chunks that represent our document the most? intuitively it's the vectors that are closest to the centroids [13]. We used clustering to get the chunks that **best represent** our document. Another way to phrase this process is, ”Which 10 documents from this book represent most of the meaning? I want to build a summary off of those.” One might argue that this causes information loss, but no summary of a whole book doesn't have information loss.

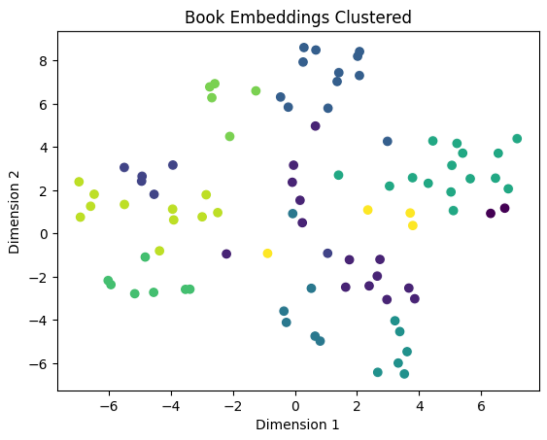

Fig. 3: We have around 70 text chunks in this example

### F. *Summarize Selected Chunks (S)*

For financial and computing reasons, We can use **gpt-3.5-turbo** for mapping and **gpt4** for reducing.

- **Mapping Prompt:**
  You will be given a single passage of a book. This section will be enclosed in triple backticks (“‘) Your goal is to give a summary of this section so that a reader will have a full understanding of what happened. Your response should be at least three paragraphs and fully encompass what was said in the passage.
  “‘text“‘
- **Reducing Prompt:**
  You will be given a series of summaries from a book. The summaries will be enclosed in triple backticks (“‘) Your goal is to give a verbose summary of what happened in the story. The reader should be able to grasp what happened in the book.
  “‘text“‘

## V. Conclusion

CRESS's potential applications extend well beyond theoretical boundaries, finding practical utility in various scenarios. It excels in creating concise audio summaries of popular books or transforming lengthy podcast episodes into shorter, more digestible versions, catering to diverse content consumption preferences. Through the integration of clustering, mapping, and reduction techniques, it paves the way for cost-effective and comprehensive content distillation. In comparison to traditional incremental updating methods, it stands out as more than ten times cheaper, making it an economically viable and sustainable solution for long document Summarization.

## References

[1] Allahyari, Mehdi, et al.“Text Summarization Techniques: A Brief Survey” arXiv preprint, arXiv:1707.02268 (2017).
[2] Liu, Yixin, et al. ”On Learning to Summarize with Large Language Models as References.” arXiv preprint, arXiv:2305.14239 (2023).
[3] Yapei, Lo, et al. ”BooookScore: A systematic exploration of book-length summarization in the era of LLMs.” arXiv preprint, arXiv:2310.00785 (2023).

[4] M. Shanahan, “Talking about large language models” arXiv preprint, arXiv:2212.03551 (2022)
[5] Zhang, Ladhak, et al. “Benchmarking Large Language Models for News Summarization” arXiv preprint, arXiv, arXiv:2301.13848 (2023)
[6] Martin Grohe ”word2vec, node2vec, graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data”, arXiv preprint, arXiv:2003.12590 (2020)
[7] Dziuba, Jarsky, et al. ”Image Vectorization: a Review”, arXiv preprint, arXiv:2306.06441 (2023)
[8] Toni Taipalus ”Vector database management systems: Fundamental concepts, use-cases, and current challenges”, arXiv preprint, arXiv:2309.11322 (2023)
[9] Zhang, Wang, et al. ”Model-enhanced Vector Index”, arXiv preprint, arXiv:2309.13335 (2023)
[10] Lin, Pradeep, et al. ”Vector Search with OpenAI Embeddings: Lucene Is All You Need”, arXiv preprint, arXiv:2308:14963 (2023)
[11] Sahoo, Paul, et al. ”The Universal NFT Vector Database: A Scaleable Vector Database for NFT Similarity Matching”, arXiv preprint, arXiv:2303:12998 (2023)
[12] Keivalya Pandya, Mehfuza Holia ”Automating Customer Service using LangChain: Building custom open-source GPT Chatbot for organizations”, arxiv preprint, arxiv:2310.05421 (2023)
[13] Bandyapadhyay, Fomin, et al. ”How to Find a Good Explanation for Clustering?”, arxiv preprin, arXiv:2112.06580 (2021)
[14] Zhou, Xu, et al. ”A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions”, arxiv preprint, arXiv:2206.07579 (2022)
[15] Luca Coraggio, Pietro Coretto ”Selecting the number of clusters, clustering models, and algorithms. A unifying approach based on the quadratic discriminant score”, arxiv preprint, arxiv:2111.02302 (2021)
[16] C.O.S. Sorzano, J. Vargas, A. Pascual Montano ”A survey of dimensionality reduction techniques”, arxiv preprint, arXiv:1403.2877 (2014)
[17] Jonathon Shlens ”A Tutorial on Principal Component Analysis”, arxiv preprint, arXiv:1404.1100 (2014)