Optimising Language Models with Advanced Text Chunking Strategies

Michael Borck

2024-04-24

This white paper explores advanced text chunking strategies and text embeddings to optimise the performance and accuracy of language models, particularly for retrieval-augmented generation (RAG) applications. It examines various chunking techniques from basic character splitting to advanced semantic and agentic methods that leverage language models to identify meaningful chunk boundaries based on content understanding. The paper provides an in-depth analysis of the RAG framework, enabling language models to search external, dynamic knowledge bases and incorporate relevant information into responses. Emphasis is placed on the subdocument RAG technique, which summarises entire documents, attaches summaries as metadata to chunk embeddings, and employs hierarchical retrieval searching summaries first for improved efficiency and contextual relevance. By combining these techniques, the paper demonstrates how language models can leverage dynamic knowledge bases to provide accurate, contextually relevant responses, paving the way for further advancements like multimodal embeddings, unsupervised chunking, adaptive chunking, incremental knowledge updates, explainable retrieval, and knowledge graph integration.

1 Introduction

As natural language processing (NLP) continues to evolve, optimising language model performance has become paramount. However, the sheer volume and complexity of textual data often hinder model performance. Text chunking strategies and text embeddings have emerged as crucial components to address these challenges.

2 Contributions and Significance

This research makes several notable contributions to the field of natural language processing and language model optimisation:

1. **Democratising Language AI with Small LLMs**: By focusing on optimising retrieval-augmented generation (RAG) and text embedding strategies for smaller, quantised open-source language models (around 4GB in size), this work aims to democratise language AI and make it more accessible to a broader range of users with limited computational resources or budgets. This approach aligns with the growing trend of developing efficient and environmentally-friendly AI solutions, reducing the carbon footprint and energy consumption associated with large-scale language models.
2. **Practical Applications and Accessibility**: Many real-world applications, such as chatbots, virtual assistants, and domain-specific language models, may not require the full capabilities of large LLMs. By optimising smaller models, this research enables practical and cost-effective solutions for a wide range of use cases, fostering innovation and inclusivity in language AI.
3. **Exploration of Novel Techniques**: The constraints imposed by working with smaller LLMs can drive innovation and potentially uncover novel techniques or insights that could be applied to larger models or inspire new research directions in the field of language AI.
4. **Reproducibility and Collaboration**: The use of open-source models and quantisation techniques promotes reproducibility and collaboration within the research community, enabling others to build upon this work and contribute to the collective advancement of language AI.
5. **Efficient and Sustainable AI**: Smaller LLMs have lower computational requirements, reducing the energy consumption and carbon footprint associated with training and deploying language models. This research contributes to the development of more efficient and sustainable AI solutions, aligning with the growing emphasis on environmental responsibility in the tech industry.

By focusing on optimising RAG and embedding strategies for smaller LLMs, this research not only expands the accessibility and practical applications of language AI but also aligns with the broader goals of developing efficient, sustainable, and inclusive AI solutions. The insights and techniques derived from this work have the potential to inspire further advancements in the field of natural language processing.

3 Retrieval-Augmented Generation (RAG)

RAG is a technique that improves the accuracy of generative models by enabling them to search through external data and fact-check before answering questions.

3.1 RAG Components

1. **Retriever**: Searches the knowledge base for relevant information using text embeddings to respond accurately to user queries.
2. **Knowledge Base**: A mutable, domain-specific knowledge base that can be updated with new information dynamically, overcoming the static knowledge limitation of traditionally trained LLMs.

3.2 RAG Implementation

1. **Preprocessing**: The RAG module preprocesses user queries by fetching relevant information from the knowledge base, enhancing the context before the query is processed by the LLM.
2. **Chunking and Embedding**: The process involves loading documents, creating embeddings, and chunking content using advanced techniques like semantic or agentic chunking to ensure only the most relevant and contextually appropriate information is passed to the LLM.
3. **Tools and Libraries**: Implementations often utilise Python libraries like Hugging Face for embeddings and indexing, as well as tools like LlamaIndex to automate chunking and summary attachment.

By combining advanced text chunking strategies, text embeddings, and RAG systems, language models can provide more accurate and relevant responses, particularly in specialised or technical domains where static knowledge may be insufficient.

4 Text Embeddings

Text embeddings are numerical representations of text that capture semantic meanings, facilitating computational processing and analysis of textual data. They enable practical applications like:

1. **Text Classification**: Training machine learning models on embedded text data to identify roles from job descriptions or detect fraudulent activities.

2. **Semantic Search**: Understanding the contextual meaning behind queries, unlike keyword search, which merely looks for exact word matches.

Text embeddings overcome the challenge of performing mathematical operations directly on text by mapping textual data into computable vectors or numbers. This allows for advanced techniques like visualising job description clusters, differentiating roles, and identifying similarities.

While large language models (LLMs) like ChatGPT can handle many text-based tasks, text embeddings are essential for specific, high-stakes applications due to lower computational costs and fewer security risks.

5 Methodology

6 Text Chunking Strategies

Text chunking involves dividing large datasets into smaller, manageable chunks, enabling language models to process information more effectively. By providing only the most contextually relevant information, chunking enhances response accuracy and relevance.

In this study, we explored and compared several text chunking strategies to enhance the performance and accuracy of language models, particularly in retrieval-augmented generation (RAG) applications. The following chunking techniques were employed:

6.1 Basic Chunking Methods

1. **Character Splitting**: Dividing text strictly by character count, which can distort words and meanings, reducing response quality.
2. **Recursive Character Splitting**: Using delimiters like new lines or specific characters to split text recursively, providing slightly more context than basic character splitting.
3. **Document-Based Chunking**: Splitting text based on document types or structures, like Python code or Markdown, aiming to retain more context compared to basic methods.

6.2 Advanced Chunking Techniques

4. **Semantic Chunking**: Using embeddings to analyse the semantic relationship between text segments, grouping chunks based on meaning and significantly improving the relevancy of data chunks.
5. **Agentic Chunking**: With this method chunks are processed using a large language model to ensure each chunk stands alone with complete meaning, enhancing coherence and context preservation.
6. **Subdocument Chunking**: The subdocument RAG technique is highlighted as an advanced strategy. It summarises entire documents, attaches the summaries as metadata to each chunk's embedding, and uses a hierarchical retrieval process searching summaries first to improve efficiency and accuracy.

6.3 Semantic Chunking

Semantic chunking of text in large language models refers to the process of breaking down long pieces of text into smaller, meaningful chunks or segments based on the semantic content and context. This is an important step when working with large language models (LLMs) that have limited context windows and can only process a certain amount of text at a time.

The key aspects of semantic chunking are:

1. It aims to divide the text into coherent and contextually relevant segments, preserving the meaning and flow of ideas within each chunk.
2. The chunking is guided by the semantic understanding of the content, rather than relying solely on predefined rules like fixed chunk sizes or separators.
3. It leverages the language understanding capabilities of the LLM itself to identify optimal break points or boundaries for chunking, based on the model's comprehension of the text's structure, topics, and semantic coherence.
4. The goal is to produce chunks that are more meaningful and contextually relevant, compared to arbitrary chunking methods. This can lead to improved performance in downstream tasks like information retrieval, question answering, or text summarisation.

The semantic chunking process typically involves:

1. Providing the full text to the LLM
2. Prompting the LLM to suggest chunk boundaries based on its understanding of the content
3. The LLM generates proposed chunk boundaries (e.g., character positions, sentence indices)
4. The text is split into chunks based on the LLM's suggestions

While semantic chunking can produce more coherent and relevant chunks, it also introduces additional complexity and computational overhead, as it requires running the LLM on the entire text corpus during the chunking process. The quality of the chunking results may depend on the specific prompting and fine-tuning of the LLM for this task.

7 Agentic Chunking

Agentic chunking is a text splitting strategy that explores the possibility of using a large language model (LLM) itself to determine how much and what text should be chunked together. It aims to leverage the LLM's understanding of the content to make more informed decisions about where to split the text, rather than relying solely on predefined rules or heuristics.

In agentic chunking, the LLM is tasked with analysing the text and identifying optimal break points or boundaries for chunking. This process typically involves the following steps:

1. The LLM is provided with the full text or document.
2. The LLM is prompted to suggest chunk boundaries based on its understanding of the content's structure, topics, and semantic coherence.
3. The LLM generates a set of proposed chunk boundaries, which can be in the form of character positions, sentence indices, or other markers.
4. The text is then split into chunks based on the LLM's suggestions.

The key advantage of agentic chunking is that it can potentially produce more semantically coherent and meaningful chunks compared to other methods like fixed-size or recursive chunking. By leveraging the LLM's language understanding capabilities, agentic chunking can better preserve the context and meaning within each chunk, leading to improved performance in downstream tasks like information retrieval or question answering.

However, agentic chunking also introduces additional complexity and computational overhead, as it requires running the LLM on the entire text corpus during the chunking process. Additionally, the quality of the chunking results may depend on the specific prompting and fine-tuning of the LLM for this task.

7.1 Subdocument RAG Technique

The subdocument RAG technique aims to improve the accuracy and efficiency of retrieval-augmented generation (RAG) systems by using document summaries to enhance the retrieval process.

Traditional chunking methods lack global context awareness, as each chunk only contains a part of a document's context. The subdocument strategy addresses this by summarising entire documents or large segments, and attaching these summaries as metadata to each chunk's embedding in the vector database.

When a user query is received, the system first searches through the document summaries to find relevant documents related to the query. It then retrieves specific chunks from these relevant documents, enhancing both efficiency by reducing the number of chunks searched, and accuracy by providing more contextual information.

This hierarchical retrieval strategy implements a two-step process: 1) Summarising and dividing documents into subdocuments, and 2) Attaching these summaries to related chunks in the database. For example, in a book on machine learning, chapters could be treated as subdocuments, with each chapter's summary aiding in retrieving relevant chunks within that chapter.

The subdocument technique is superior to basic RAG setups without summaries, as it improves response times, relevance, and overcomes the lack of global context in traditional chunking methods. Tools like LlamaIndex simplify implementation by automating the chunking and summary attachment process.

Yes, it would be appropriate to include a section or subsection describing the evaluation methods used to assess the performance and effectiveness of the different text chunking strategies discussed in the white paper.

Here's how you could structure this section:

8 Evaluation Methods

To comprehensively evaluate the various text chunking strategies and their impact on language model performance, we employed the following evaluation methods:

8.1 Chunk Coherence and Relevance

1. **Coherence Scores**: We utilised topic modelling algorithms to compute coherence scores for the chunks produced by each chunking strategy. Higher coherence scores indicate that the chunks are more semantically coherent and contextually relevant.
2. **Human Evaluation**: A subset of the generated chunks was manually evaluated by human annotators, who assigned scores based on the quality, meaning preservation, and contextual relevance of the chunks.

8.2 Downstream Task Performance

The effectiveness of the chunking strategies was further assessed by evaluating the performance of language models on downstream tasks when using the generated chunks:

1. **Question Answering**: We measured the accuracy and F1 scores of language models on question-answering tasks, using chunks from different chunking methods as input.
2. **Information Retrieval**: The chunks were used to build retrieval systems, and we evaluated the precision, recall, and mean reciprocal rank (MRR) of these systems on relevant information retrieval benchmarks.
3. **Text Summarisation**: We assessed the quality of summaries generated by language models when provided with chunks from various chunking strategies, using metrics like ROUGE scores and human evaluation.

8.3 Efficiency Metrics

To understand the computational efficiency and scalability of the chunking strategies, we measured and compared the following metrics:

1. **Chunking Time**: The time taken by each chunking strategy to process a given corpus or dataset, plotted against the dataset size to assess scalability.
2. **Memory Usage**: The memory footprint of each chunking strategy during the chunking process, which is crucial for resource-constrained environments or large-scale applications.

By combining these evaluation methods, we aimed to provide a comprehensive analysis of the strengths and weaknesses of each text chunking strategy, considering both the quality of the generated chunks and their impact on downstream language model performance, as well as the computational efficiency and scalability of the approaches.

9 Future Work

While the techniques discussed in this paper have shown promising results in enhancing the performance and accuracy of language models, there are several areas that warrant further exploration and research:

1. **Multimodal Embeddings**: Extending text embeddings to incorporate multimodal data, such as images, videos, and audio, could unlock new applications and improve the contextual understanding of language models.
2. **Unsupervised Chunking**: Developing unsupervised methods for text chunking that can automatically identify optimal chunk boundaries without relying on labeled data or human intervention could lead to more scalable and efficient solutions.

3. **Adaptive Chunking**: Exploring adaptive chunking strategies that can dynamically adjust the chunk size and granularity based on the complexity of the content or the specific task at hand could further improve the trade-off between computational efficiency and contextual relevance.
4. **Incremental Knowledge Base Updates**: Investigating techniques for seamlessly integrating new information into existing knowledge bases without the need for complete retraining or rebuilding could enhance the adaptability and scalability of retrieval-augmented generation systems.
5. **Explainable Chunking and Retrieval**: Developing methods to provide interpretable explanations for the chunking and retrieval decisions made by language models could increase transparency and trust in these systems, particularly in high-stakes applications.
6. **Integration with Knowledge Graphs**: Investigating the integration of structured knowledge graphs with retrieval-augmented generation systems could unlock new possibilities for leveraging structured data and enhancing the contextual understanding of language models. Techniques for seamlessly incorporating knowledge graph information into the retrieval and generation processes could be explored.

10 Conclusion

The integration of advanced text chunking strategies and text embeddings has proven to be a powerful approach for enhancing the performance and accuracy of language models, particularly in retrieval-augmented generation applications. By effectively managing and processing large volumes of textual data, these techniques enable language models to provide more relevant and contextually appropriate responses.

The subdocument RAG technique, which leverages document summaries to improve retrieval efficiency and contextual relevance, has emerged as a promising solution for overcoming the limitations of traditional chunking methods. By combining advanced chunking techniques with text embeddings and the RAG framework, language models can better leverage dynamic, domain-specific knowledge bases, leading to more accurate and reliable responses.

As natural language processing continues to evolve, further research and development in areas such as multimodal embeddings, unsupervised chunking, adaptive chunking, incremental knowledge base updates, and explainable chunking and retrieval will be crucial for unlocking the full potential of these techniques and addressing the ever-increasing complexity of textual data.