ASCENT

How does Azure AI Document Intelligence boost RAG performance?

Insight Guide.

Andrew May

Introduction.

Retrieval-Augmented Generation (RAG) systems enable businesses to combine their proprietary data with the power of AI to generate more accurate and contextually relevant responses. While they often appear as easy-to-use interfaces like chatbots, there's a lot more going on under the hood. These systems rely on a suite of advanced tools for tasks including data ingestion, preprocessing, and context retrieval, all of which can be optimised to improve the quality of answers provided by a large language model (LLM).

Here, we explore how Azure AI Document Intelligence can enhance the performance of a RAG pipeline by introducing features such as **semantic chunking** and **text extraction from images and tables**. We've conducted a study to evaluate the impact this addition has on the quality of LLM-generated answers, compared with using a more standard chunking approach.

About the author.

Andrew May

Andrew has worked in AI & Data Science for 3+ years, applying Python and Azure-based solutions to a range of public and private sector challenges, including central government, defence, and insurance. Having completed a PhD in Physics from Cardiff University, Andrew combines analytical rigour and practical implementation skills to help our customers achieve real-world business value using cutting-edge AI technologies.

What is standard chunking and why is it necessary?

Chunking **divides large texts into smaller segments** (chunks). When a RAG system receives a prompt, relevant chunks are retrieved to help inform (augment) the LLM response. Chunking is essential for RAG systems because it ensures that the text fits within the input constraints of LLMs and it enables the efficient search and retrieval of relevant information, improving the accuracy and relevance of generated responses.

Standard chunking splits the original text based on a **fixed token count**. This approach is simple and convenient to implement, although it doesn't consider the semantic coherence of the content, which can lead to issues such as a chunk ending mid-sentence, potentially losing context and limiting performance for complex queries.

What is semantic chunking and how does Document Intelligence implement it?

Semantic chunking groups text into segments based on their semantic subject rather than simple token count. This method uses complex algorithms so is far more computationally intensive, however, it enhances the relevance and coherence of information retrieval by enabling each chunk to maintain semantic consistency as well as to retain complete and meaningful information.

Azure AI Document Intelligence is a cloud-based service for building intelligent document processing solutions. Document Intelligence implements semantic chunking using a layout model, this facilitates text ingestion from various source formats (docx, xlsx, PDF, etc.) and outputs the text into structured Markdown (see Figure 1). Chunks are then convenient to create by splitting the text on Markdown headers. Additionally, Document Intelligence extracts text from images and tables, expanding the scope of accessible data.

Figure 1: Text image processed with Document Intelligence Studio (left) and output to MarkDown using Layout model (right)[1].

[1]Source: https://learn.microsoft.com/en-us/azure/ai-services/document-intelligence/concept-retrieval-augmented-generation?view=doc-intel-4.0.0#semantic-chunking-with-document-intelligence-layout-model

Comparison of RAG performance using standard and semantic chunking.

To understand the benefit of this Azure AI Service, we ingested 122 internal policy documents to create 2 independent indexes, one using standard chunking and the other using semantic chunking via Document Intelligence. We then used each index for the context retrieval step of a RAG system to generate answers from a manually created evaluation set of 30 questions. Once generated, the 2 sets of answers were quantitatively compared to evaluate the impact of Document Intelligence.

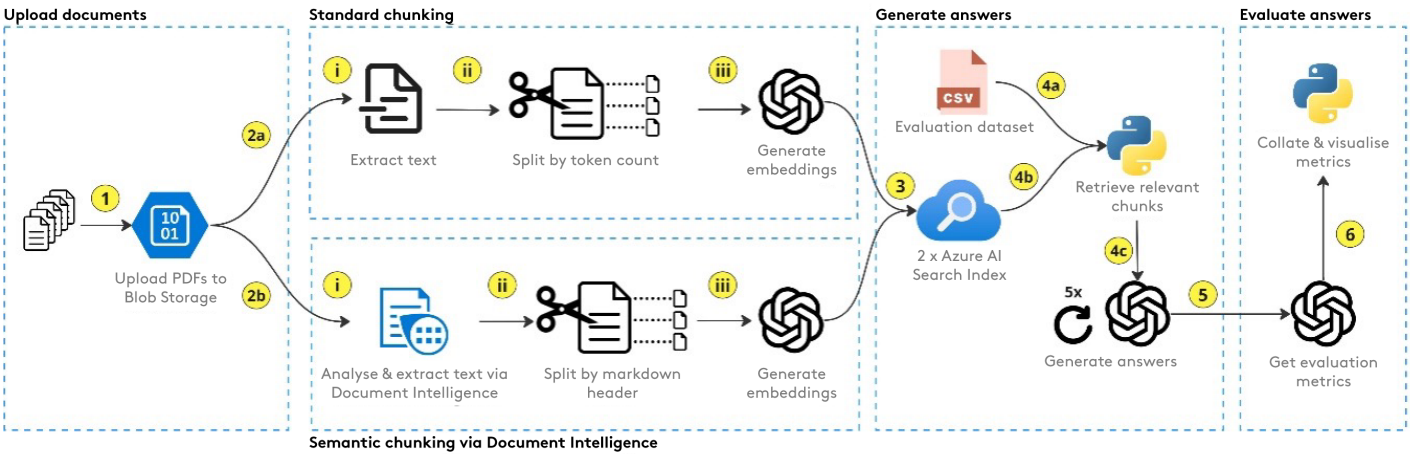

Figure 2: Process diagram for the comparison of standard and semantic chunking.

Document ingestion and chunking.

The workflow began by uploading the PDF documents to Blob Storage (1), triggering 2 separate chunking methods to process the documents. The 2 chunking methods are:

— **Standard chunking (2a):** This method extracts the text from a PDF (i), the text is then split into chunks by token count with an upper size limit of 500 tokens (ii), before vector embeddings are generated from each chunk (iii).

— **Semantic Chunking via Document Intelligence** (2b): Here the PDFs are passed to Document Intelligence (i) which analyses the document structure and extracts the text into clusters of similar semantic meaning, which are outputted in Markdown format (see Figure 2). This Markdown text is then divided into chunks using headers as delimiters, before vector embeddings are generated from each chunk (iii).

Generating answers from both indexes using RAG.

Every document was ingested by both chunking methods to create 2 separate indexes - hosted in Azure AI Search - that could be compared (3). To understand the benefit of Document Intelligence, a RAG pipeline was employed to answer an evaluation set of questions using both AI Search indexes. Hence, the quality of the generated answers indicates the relative benefit of the 2 chunking techniques. The RAG pipeline:

— Ingested an evaluation set of 30 questions and ground truths related to Ascent's company policies (4a). Embeddings were generated from each question using Azure OpenAI Service.

— Performed a vector search to retrieve the most relevant chunks for each question from both indexes to define a context (4b).

— Generated answers by prompting an LLM with each question and the retrieved context (4c). This was repeated 5 times for each question with a temperature of 0.3, leading to **150 LLM responses** for **30 unique questions**.

Evaluating the question answering performance.

Traditional methods for determining machine learning evaluation metrics are typically applied to discrete outputs, however the answers generated here are qualitative. For example:

Question: *"What are Ascent's core values?"*

Answer: *"Ascent's core values are Energy, Empathy, and Audacity. These values are integrated into the company's culture and are used to measure success. They are reflected in the way employees treat each other and are incorporated into self-assessments and appraisals."*

One method for quantitative evaluation of qualitative data is to prompt another LLM with the definition and scoring criteria of an evaluation metric, such as precision and recall. Next, ask the LLM to return a discrete score by comparing the LLM-generated answer with the ground truth (5).

Precision typically answers the question "of all the instances that the model predicted as positive, how many were actually positive?". This can be re-written to ask, "of all the claims made in the LLM-generated answer, how many of them are found in the ground truth?". Similarly, **recall** can be defined as "of all the claims made in the ground truth, how many of them are found in the LLM-generated answer?". The evaluation prompt then requires the LLM to return a score of 1, 2, 3, 4, or 5, and gives defines the expectation for each score.

Python scripts orchestrated this evaluation by submitting prompts and collating evaluation metrics for analysis (6). Precision and recall scores were obtained for all the 150 LLM responses to 30 unique questions through prompting, F1 scores were then subsequently calculated.

Evaluation Results.

The evaluation metrics are visualised in Figure 3, where the bar height indicates the (mean score ± standard deviation) across all 150 LLM responses. Precision, recall, and F1 Score all show an **8% increase in performance** when the context retrieval step is performed with the index created by Document Intelligence, when generating answers for this evaluation set of 30 unique questions. For all 3 metrics, this enhanced performance is proven to be statistically significant using a two-sample z hypothesis test with a α threshold of 0.01.

Document Intelligence yields 8% increase in all metrics.

Based on 150 LLM responses

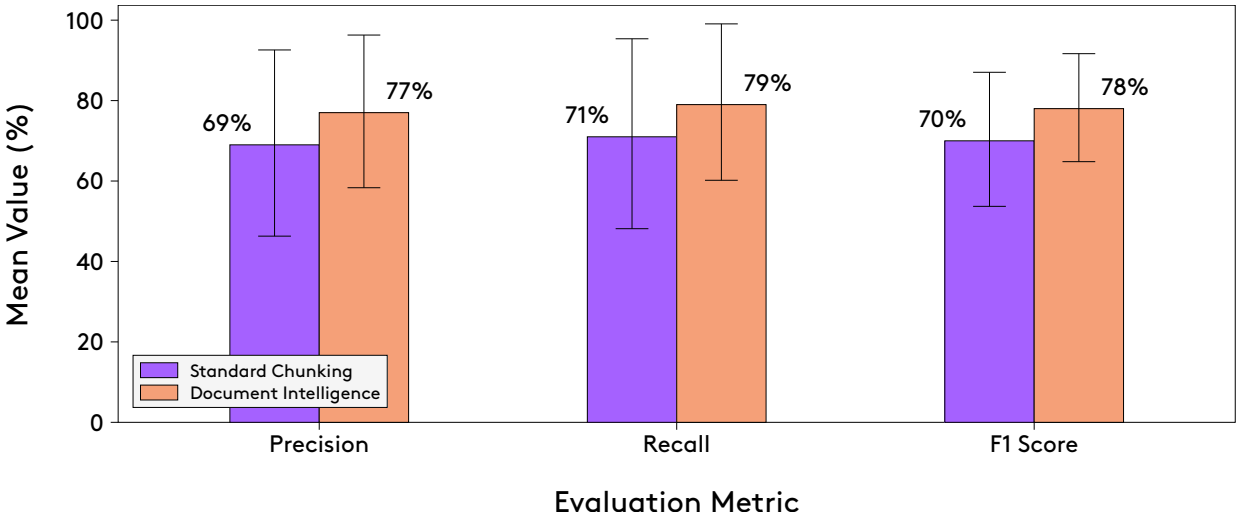

Figure 3: Evaluation of LLM-generated answers. Context retrieval was separately performed using indexes made with standard chunking & semantic chunking via Document Intelligence.

Conclusion.

In this study, integrating Azure AI Document Intelligence into a RAG pipeline has shown **clear benefits** for the quality of LLM-generated answers. This approach improves the quality of responses by:

— Grouping segments of text into chunks with more similar semantic content.

— Extending the scope of information retrieval by extracting text from tables and images into Markdown.

For businesses, this means a more reliable system capable of extracting and presenting information more effectively, leading to improved decision-making and customer satisfaction. As we continue to explore innovative ways to augment AI with custom data, tools like Document Intelligence will play a key role in pushing the boundaries of what's possible.

Explore how RAG-based conversational systems can revolutionise your website's user experience with enhanced text and visual data. Let's get started!

ascent.io/services/data-ai/artificial-intelligence

ascent.io