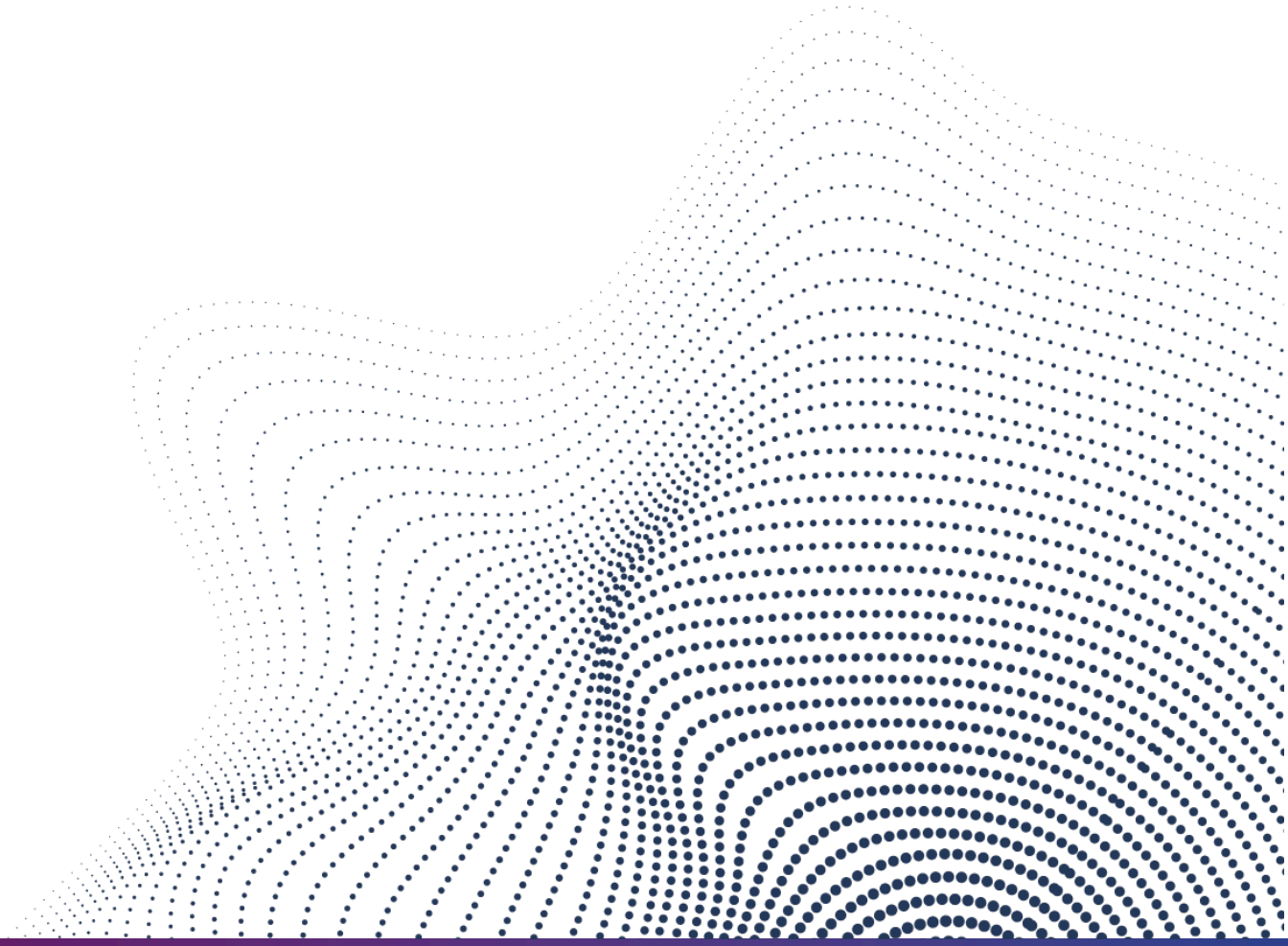
2025

CableLabs®

On Writing for AI

Best Practices and Guidelines for Creating AI-Ready Documents for AI Processing



Revised

April 29, 2025

Marcos Gava, Sentosa Technology Consultants
marcos@sentosatech.com

Austin Pahl, CableLabs
a.pahl@cablelabs.com

Megan Collins, CableLabs
m.collins@cablelabs.com

Contents

Introduction

This document provides best practices and guidelines for creating documents that are optimized for seamless integration with Artificial Intelligence (AI) systems, particularly those leveraging large language models (LLMs) and Retrieval-Augmented Generation (RAG) techniques. Adhering to these guidelines will significantly improve the accuracy, efficiency, and overall effectiveness of your team's workflows when using AI for information retrieval and analysis. This guide focuses on preparing documents for optimal embedding and processing by AI models, ensuring reliable and consistent results.

Basic Concepts

When you type a message into a chat system like ChatGPT, a series of events occur before your message is “read” and processed by a large language model (LLM). Here is a quick summary of what happens:

1. As you type the message, it is typically stored as a sequence of Unicode characters.
2. Once the message is submitted, the message text is broken into a series of “tokens” which are the smallest chunks of information that a language model is trained to understand. An example of tokenization can be found here.
3. The series of tokens is converted into a series of embedding vectors, which are essentially numerical descriptions of the semantics associated with each token.
4. The embedding sequence is provided to the LLM as input.

When an LLM is finished processing inputs, it outputs embeddings that are then converted to human-readable text by following the above steps in reverse. This output is the “response” that ChatGPT/etc. provides to the user based on their input message.

As described above, the first thing that needs to be provided to this type of AI system is machine-readable text, often formatted as Unicode (or ASCII). If you see an AI application that can read uploaded documents, that application has some method of recovering machine-readable text from the input file. Some document file formats contain the full, unaltered text and software can automatically extract this content with ease. In other cases, such as a physical document that was scanned into a digital image, optical character recognition (OCR) must be used to recover the text (or an approximation thereof).

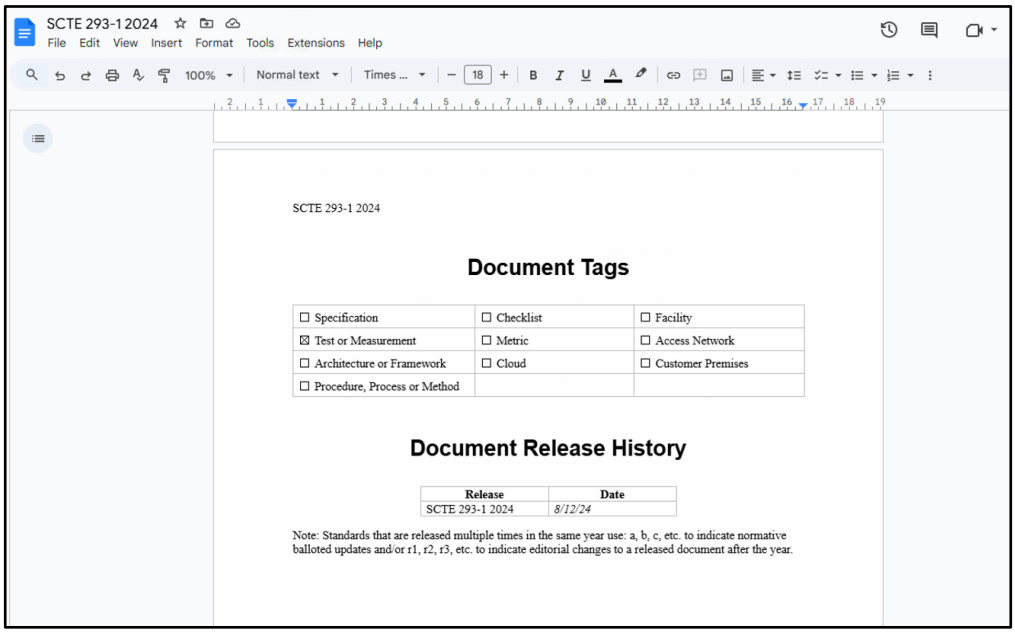
The message at the heart of this paper is that the most AI-ready documents are the ones that have a complete representation of their content as machine-readable text that can be automatically extracted without error. The following guidelines focus on supporting this goal.

Writing AI-Ready Content

Use Semantic Document Structure

Use headings, lists, tables, and other structural elements liberally and consistently. This allows AI models to better understand the hierarchical structure and relationships within your document.

- **Example:** *Many Word features like numbered lists, bulleted lists, tables, headers, and footers are useful structures for supporting AI readability in text. For example:*



1. Screenshot of a document containing a table, headers, and special characters

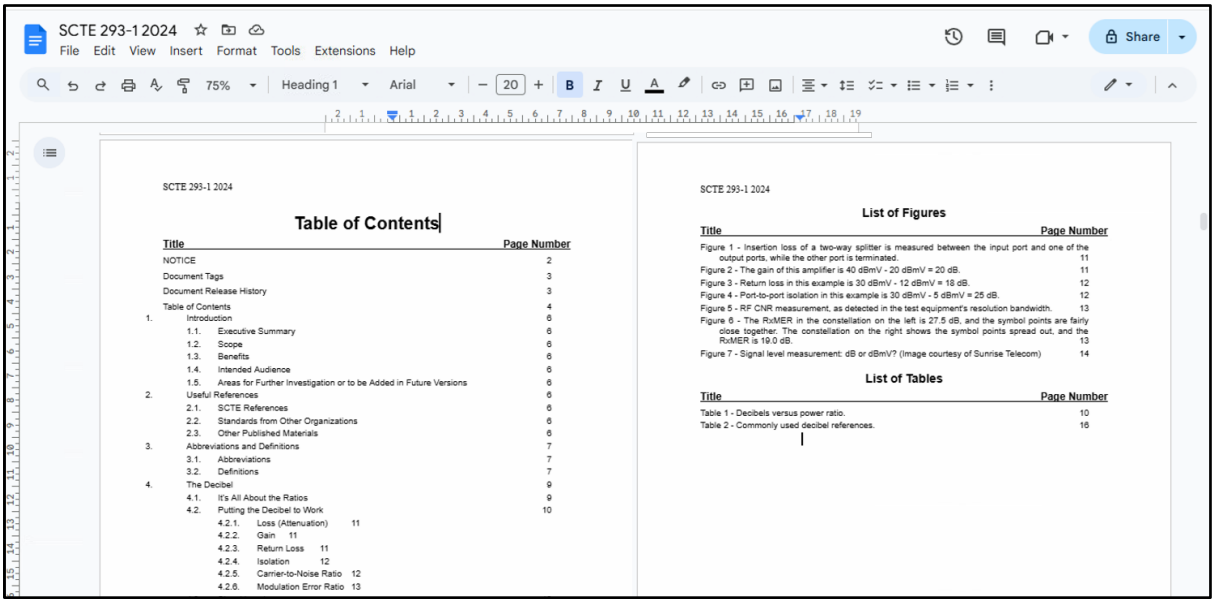
```
== Document Tags

[width="100%",cols="36%,31%,33%",options="header",]
|===
|☐ Specification |☐ Checklist |☐ Facility
|☒ Test or Measurement |☐ Metric |☐ Access Network
|☐ Architecture or Framework |☐ Cloud |☐ Customer Premises
|☐ Procedure, Process or Method | |
|===

== Document Release History

[width="100%",cols="50%,50%",options="header",]
|===
|Release |Date
|SCTE 293-1 2024 |_8/12/24_
|===

Note: Standards that are released multiple times in the same year use: a, b, c, etc. to indicate normative balloted updates ar

== Table of Contents
```

2. Screenshot of an AsciiDoc document converted from Figure 1, with most of the syntax preserved and therefore visible to AI applications

Provide Clear Navigation

Structure documents logically to aid navigation and context retrieval for AI processing. Use logical document flow, clear section and subsection headings, and clear image and table captions.

- **Example:** *Organize information in a logical sequence, with each section building on the previous one.*



3. Example of document structure

Prioritize Meaningful Content, Avoid Redundancy

If this doesn't interfere with the intended human use of the document, ensure as much of the textual content is unique and semantically meaningful. That is, avoid including long asides or content that is intended to be humorous to the audience but does not contribute to the meaning of the document. Strive to write with clarity and conciseness over verbosity and human enjoyment. Restating, paraphrasing, using asides, and including unnecessary content frequently hinders ability of RAG systems and citation algorithms to achieve accurate results. In RAG systems, this results in inaccurate or irrelevant responses to prompts and in citation algorithms, it causes incorrect attribution of the LLM's response to source content.

Prefer Simple Tables

Structure tables simply, using clear headers and row/column labels. Avoid complex nested tables or merged cells, as this hinders accurate data extraction and can confuse LLMs.

- **Example:** *Use simple grid-like tables with a header row and clear labels for each column. Avoid using merged cells or nested tables.*

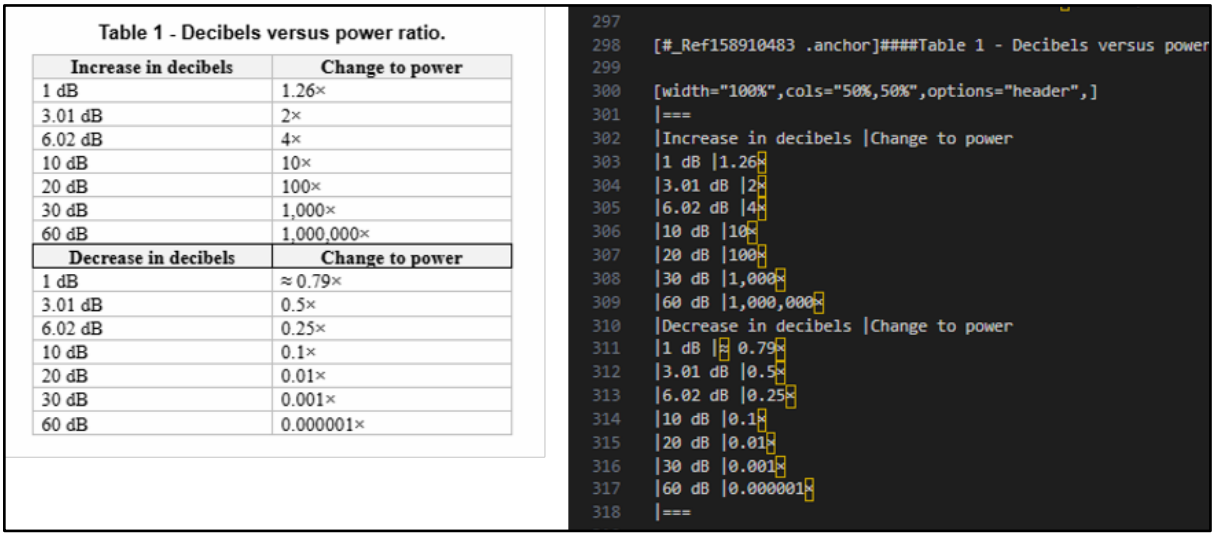


Table 1 - Decibels versus power ratio.

Increase in decibels	Change to power
1 dB	1.26×
3.01 dB	2×
6.02 dB	4×
10 dB	10×
20 dB	100×
30 dB	1,000×
60 dB	1,000,000×
Decrease in decibels	**Change to power**
1 dB	≈ 0.79×
3.01 dB	0.5×
6.02 dB	0.25×
10 dB	0.1×
20 dB	0.01×
30 dB	0.001×
60 dB	0.000001×

```
297
298 [#_Ref158910483 .anchor]###Table 1 - Decibels versus power
299
300 [width="100%",cols="50%,50%",options="header",]
301 |===
302 |Increase in decibels |Change to power
303 |1 dB |1.26×
304 |3.01 dB |2×
305 |6.02 dB |4×
306 |10 dB |10×
307 |20 dB |100×
308 |30 dB |1,000×
309 |60 dB |1,000,000×
310 |Decrease in decibels |Change to power
311 |1 dB |≈ 0.79×
312 |3.01 dB |0.5×
313 |6.02 dB |0.25×
314 |10 dB |0.1×
315 |20 dB |0.01×
316 |30 dB |0.001×
317 |60 dB |0.000001×
318 |===
```

4. Left: A table from a Word document. Right: the same table converted into the AsciiDoc format.

When considering ways to represent tables in plaintext, prefer formats that are well represented in LLMs' training data. Markdown is extremely common and well-understood by LLMs. On the other hand, Markdown table syntax is very limited – for more complex table formatting, AsciiDoc can represent details like merged cells and nested tables, so that data may be preserved (although it's still harder for LLMs to understand these details in practice).

When adding tables to documents, use Word's Insert Table functionality to create tables within the document. Avoid creating your table as an image file or screenshotting an image from a different document and inserting the tables as an image within the document. This will prevent any of the content of the table from being extracted during the document conversion process, which decreases the LLM's ability to accurately answer questions about content contained in those tables.

Creating AI-Ready Visuals

Describe Images Using Alternative Text

Provide comprehensive, descriptive alternative text (often called "alt text" in short) for all images. This allows AI models (and screen readers) to understand the content and context of the image, even if the model cannot "see" the image like a human.

Deciding what to write for alt text is subjective. A general rule of thumb can be to write enough that a screen reader can convey to its user the *intended message* of the figure in the context of its section and the overall document. In some cases, this can be challenging – trying to completely describe a large, detailed flowchart or architecture diagram with alt text alone is impractical. In those cases, alternative methods may be explored: one way is to rewrite the content around the diagram to describe it in full detail, or another way is to use a diagram description language like Mermaid to provide a code-like description of the image. LLMs have been shown to do well at understanding Mermaid diagrams, but it still is not a definitive solution in all cases.

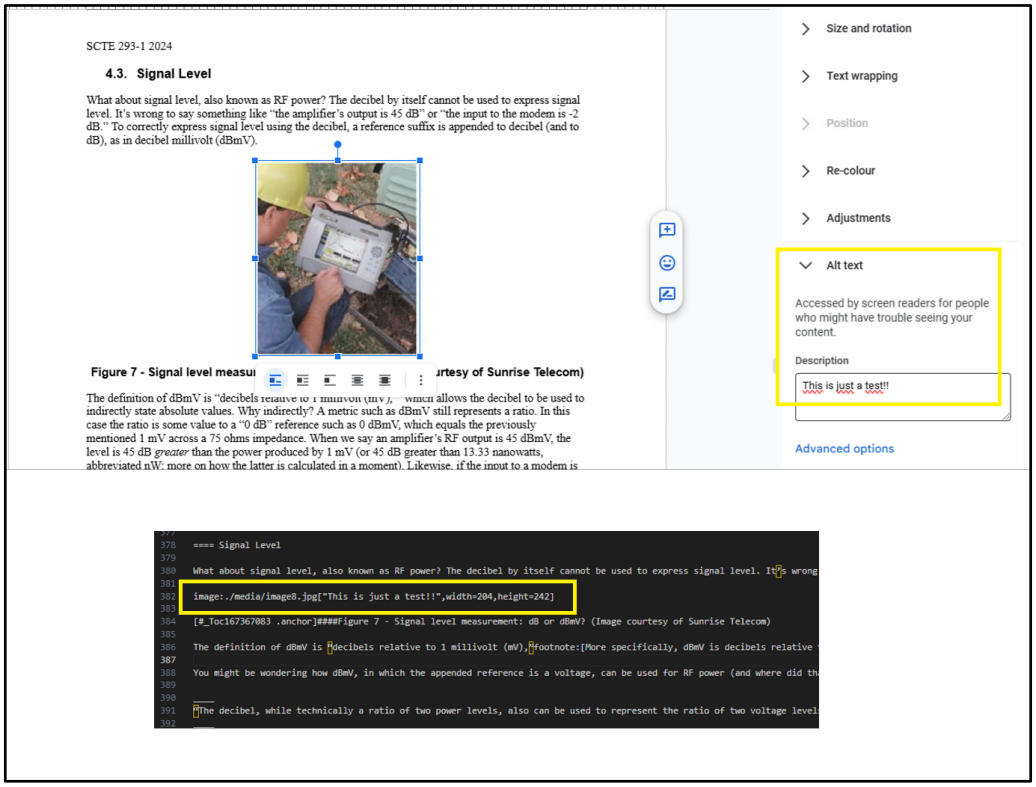
Vision language models (VLMs) and commercial LLMs that provide support for image processing can be used to generate alt text for images within a document. While human-generated alt text is always preferred, VLM-generated text provides an excellent starting point for alt text. From our experiments, we found that VLMs including Mistral's pixtral model and Google's Gemini 2.0 Flash model provide excellent contextualized summaries of images from various documents.

The EU's alt text guidelines and guidelines from institutions supporting the visually impaired recommend limiting alt text to 125 characters. Additionally, alt text descriptions should describe the context of the image, particularly the intention behind including that specific image within the document. Alt text should not be provided for decorative images or logos and default alt text generated by Word should be removed for those images.

From our experiments, limiting alt text to 125 characters severely limits the ability of LLMs to answer questions related to images found within the documents. We therefore recommend that documents which will be read and utilized by humans use alt text descriptions that adhere to the EU's guidelines to enhance screen reader accessibility, while documents that will be integrated into RAG systems have comprehensive alt text added to each of their figures. In situations where maintaining multiple copies of documents is infeasible, we recommend rewriting the text around images to fully explain the context of the image and utilize alt text to provide a high-level summary of the image's content and its context.

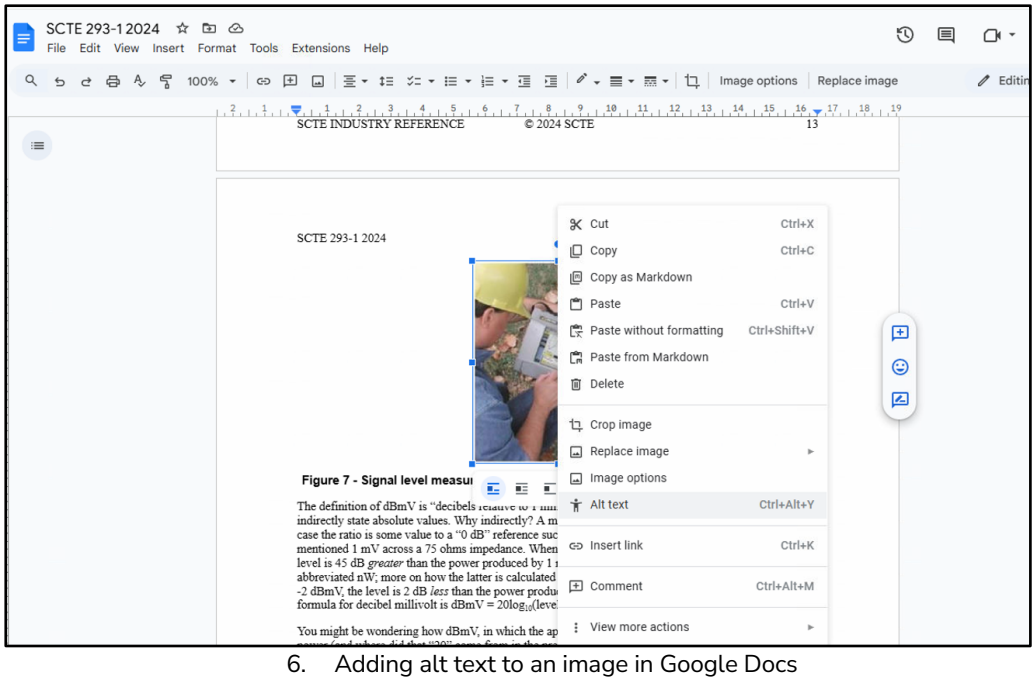
Figure descriptions may also be used within documents to provide a longer, more comprehensive description of images found within the document, while ensuring that the alt text for those images adheres to the EU's guidelines. These descriptions are usually found immediately following the image and provide a longer description of the image's content and context in the same way as rewriting the text surrounding the image. The generation of figure descriptions can similarly be automated using VLMs.

- **Example:** *Instead of "image1.png", use "A bar chart showing quarterly sales figures for 2023."*



5. Example of alt text

- **Note:** Adding alt text to an image is supported in most document editors. In Microsoft Word and Google Docs, right click an image and select the "Alt text" option as shown below.



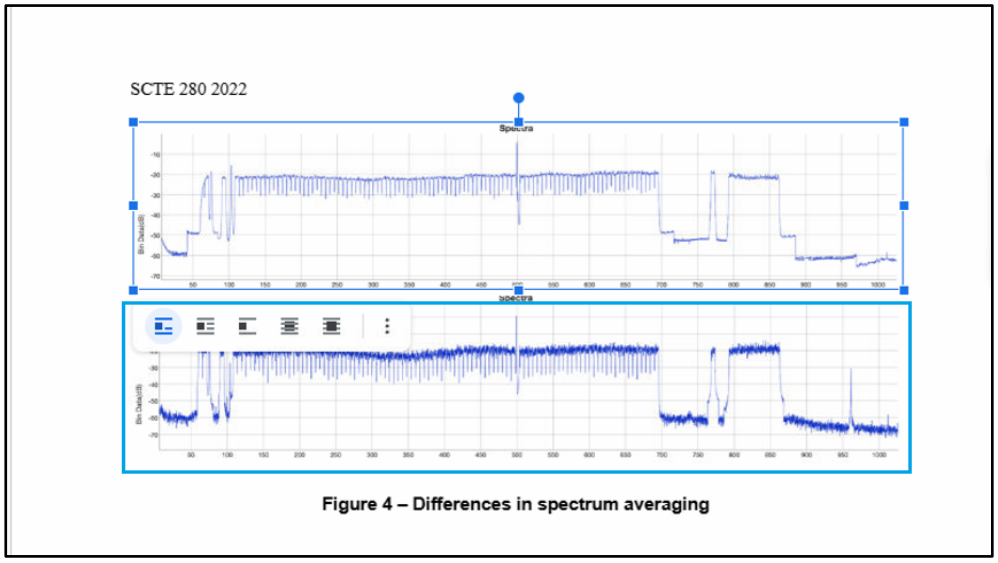
6. Adding alt text to an image in Google Docs

Prefer Simple Image Formats

When embedding images in a document, use either PNG or JPEG image files, since these file formats support both document conversion, VLM-generated alt text, and multimodal RAG. . Also, these formats are the ones where most document applications (MS Word, Google Docs, etc.) allow applying alt text, which can be invaluable for AI readability.

If there are multiple image files that will be referenced with a single caption, as occurs with a grid of images or stacked image files that share a caption, the image files should be merged into a single image file before being inserted into the document. During document conversion, each image file will be extracted from the document separately, which prevents the LLM from recognizing that multiple image files are related to one another and share a caption.

- **Example:** Two stacked images that share the same caption, “Figure 4”; this should be avoided by merging the image files into a single image, which enables the LLM to recognize the relationship between the images.



7. Example of a composite image

For adding images to documents, do not use built-in art, shape, text box, word art, chart, or icon features from your document editor (Word, Google Docs, etc.) to create or augment image files within the document. Instead, insert your image files by uploading or inserting them from files to ensure that their file type is not altered during the insertion process. Using other methods to insert images into the document or creating/augmenting image files using built-in functionalities prevents the document conversion from recognizing those images as image files. Additionally, document editors will not provide the option of adding alt text to built-in objects, so alt text would need to be manually added to the converted document using Markdown or Asciidoc.

Handling AI-Ready Files

Convert to MD or ADOC, not Unformatted Text

When converting a document into a text representation, prefer a widely used markup language like Markdown (.md extension) or AsciiDoc (.adoc extension). This requirement ensures consistent character encoding, avoids compatibility issues across different platforms and AI/LLM tools, and provides a standardized format that is easily parsed by AI systems. Most LLMs recognize Markdown and AsciiDoc, and can understand meaningful structural elements like headers, lists, and tables. In contrast, nonstandard text conversion tools do not preserve these elements (or may do so in an unfamiliar way).

Converting from DOCX

If documents are produced in Microsoft Word/Google Docs/etc. and have been produced adhering to the guidelines listed above for image formats and the curation of alt text, the best method for converting those documents to machine-readable formats is using Pandoc. Pandoc and distributed as PDF, avoid using the PDF as the basis for conversion to plaintext. Prefer the authoring format instead (e.g. DOCX). PDFs present inherent limitations in fidelity during text and media extraction, often resulting in data loss, formatting issues, and challenges with complex layouts.

Anecdotally, we faced considerable challenges around data loss and formatting when converting from PDF to text early in our work. As a result, we switched to a DOCX to AsciiDoc/Markdown workflow to improve fidelity and efficiency, significantly reducing errors and improving data integrity.

- **Example:** *After finalizing your content in DOCX, use a conversion tool like Pandoc to create an AsciiDoc file (.adoc extension).*

Converting from PDF

If documents are only available in PDF format or if they were produced in Microsoft Word/Google Docs/etc. but did not adhere to the guidelines listed above for image formats and the curation of alt text, then the best method for converting those documents is to use an OCR solution. Note that non-PDF documents meeting the criteria for OCR conversion must be converted to a PDF format before being converted using OCR. While historical OCR solutions have performed poorly, recent research has significantly improved the performance of modern OCR solutions, many of which incorporate AI models to further improve the extraction of text and structure from documents.

During our experiments, we saw high accuracy when converting PDF documents to Markdown using Mistral's OCR solution, Google's Gemini 2.0 Flash model, IBM's Docling models, and the RolmOCR model which is an adapted version of AI2's olmOCR model. From our experiments, Mistral's OCR solution and Google's Gemini 2.0 Flash model both had

excellent accuracy; Docling and RolmOCR had lower accuracy but are locally hosted, open-source models which ensures that confidential and proprietary data is never sent off-premises for conversion.

While the resulting Markdown file from OCR conversion can be used in a RAG system, we saw optimal results when converting the Markdown file to an Asciidoc file using Pandoc. Additionally, Docling and Mistral's OCR models facilitate the easy addition of alt text to the Markdown document; Gemini and RolmOCR can be augmented with alt text, but the process of adding alt text to those Markdown files is more tedious.

Track Source Materials Used to Produce the Document

Often tools are used to generate visuals used in technical documents. For example, we have seen Visio-based flowcharts and diagrams, and we've seen line plots from real measurements gathered in the field. In either case, it is valuable to keep the source content that was used to generate the visuals (Visio project files and raw data files, respectively).

The source content is most useful for using VLMs to generate an initial version of alt text which can then be augmented by a human to ease the process of generating alt text for a large corpus of documents. This process requires the original image files used in the documents in either PNG or JPEG format to send those image files through the VLM and generate initial alt text for each of those image files.

Tool use is a software design pattern where LLMs can be made to utilize alternative data formats like these as sources of context. Additionally, the rapid evolution of capabilities in VLMs has enabled significant improvements in multimodal RAG, which enables VLMs to consider the full content of the image when answering questions about that image. Multimodal RAG also includes the context around the image when sending a question to the VLM about a specific image or section of the document, enabling the VLM to access the full content that would be available to a human when reading the document. Similar to generating alt text, multimodal RAG requires that the image files from each document used in that RAG layer be available to the VLM in either PNG or JPEG format.

Also, keeping the source materials is useful for long term maintenance of the documents in case visuals need to be updated for any reason in the future.

Additional Commentary

Accessible Documents are AI-Ready Documents

During our initial pilot projects, accessible design principles dramatically improved the accuracy of information extracted by our RAG system. Specifically, the consistent use of alt text for images allowed the AI to accurately answer questions about visual content, particularly when compared to documents without alt text.

Accessibility is a worthy goal regardless of AI, but achieving accessibility also effectively satisfies the requirements needed to make a document useful to AI. As mentioned previously, there are different considerations for generating alt text that is accessible and useful to LLMs and generating alt text that is useful and accessible to screen readers. When designing documents to be accessible to humans, we recommend adhering to the guidelines provided by the EU for alt text curation. Additionally, we recommend referencing and incorporating the advice from Perkins School for the Blind and American Foundation for the Blind.

Further reading:

- Microsoft Support: *Make your Word documents accessible to people with disabilities*
- Section508.gov: *Create Accessible Documents*
- Perkins School for the Blind: How to Write Alt Text for the Visually Impaired
- American Foundation for the Blind: Writing Alternative Text
- EU Alt Text Guidelines: Alternative Text

Consistency is Key

Maintain consistency in formatting, structure, and terminology throughout your documents. This is relevant at all layers of document production – make sure document files are named consistently and predictably, make sure document metadata like publish dates and revision names are presented in the same way, in the same place across all documents in a dataset, and make sure document structure is clear and uniform.

Test Your Documents Thoroughly and Often

Before deploying documents for AI processing, test them using the target AI system to ensure they perform as expected. Tests should be reproducible and easy to run often. At a minimum, identify high-priority information in source documents and ensure that retrieval of that information is tested to behave correctly.

To illustrate one possible solution for testing, the following sections describe the initial methods we took to test our AI applications. Our approach focuses on creating structured Q&A pairs and leveraging an automated testing platform to ensure thorough and consistent evaluation.

Q&A Pair Methodology

At the core of our testing process is the creation of targeted Question and Answer (Q&A) pairs. Each Q&A pair consists of a question designed to assess the LLM's understanding of a specific concept or relationship within the document, along with a carefully crafted expected answer. By comparing the LLM's response to the expected answer, we can quantitatively measure the accuracy of the model's understanding.

Our Q&A pair generation process currently employs a blended approach. To ensure a solid foundation of fundamental knowledge assessment, a significant portion of the questions are

created manually. These manually crafted questions are often more conceptual in nature, focusing on explicit definitions and relationships that are directly stated within the documents. This allows us to directly assess the LLM's ability to extract and recall key information.

To supplement the manual question generation, we leverage AI models, currently GPT-4o Mini, to create additional Q&A pairs. This approach allows us to explore more complex and nuanced aspects of the documents. AI can be used to generate questions that combine concepts from different sections, explore implicit relationships, or assess the LLM's ability to synthesize information. Furthermore, we ensure that all AI-generated questions are designed to produce answers that can be directly derived from the provided document, maintaining the integrity of the evaluation.

The responses to these Q&A pairs are then scored according to answer correctness. This involves assessing the accuracy of the LLM’s responses by comparing them to the pre-defined expected answers. This metric quantifies the accuracy of the LLM’s responses. We determine the percentage of answers that directly align with the expected answer, based on the information provided in the documents. A higher percentage indicates greater accuracy.

Automated Testing Platform

To streamline the testing process and enable efficient experimentation with various LLMs, evaluation methodologies, and RAG implementations, we utilize an automated testing platform that is available as open source. This platform provides the following key functionalities:

- **Automated LLM Evaluation:** The platform allows us to run batch jobs that test LLMs against a series of pre-defined questions, automatically scoring them against the expected answers. This significantly reduces the manual effort required for evaluation and enables rapid iteration and comparison of different configurations.
- **Flexible Evaluation Settings:** The platform offers a range of configurable settings to tailor the evaluation process. This includes:
    - **Evaluation Type:** The ability to select different methods for evaluating LLM responses, including but not limited to metrics from Ragas.
    - **Number of RAG Runs:** The ability to specify how many times to test the questions using context from RAG.
    - **Number of Non-RAG Runs:** The ability to specify how many times to test the questions without context from RAG.
- **Comprehensive Reporting and Export:** Once the execution has completed, the platform provides a summary of the results, including key metrics such as accuracy, precision, and recall. The results can be exported to JSON format for further analysis and integration with other tools.
- **Leaderboard Integration:** A leaderboard interface is provided for easy comparison of

different LLMs and RAG configurations.

This automated testing platform offers a valuable framework for efficiently evaluating and optimizing LLM performance in the context of technical documentation processing.

Conclusion

By following these guidelines, your team can generate AI-ready documents that consistently deliver reliable results during RAG processing. This will directly improve your workflow, reduce manual errors, and facilitate more intelligent and efficient utilization of AI technologies. This document will be updated as practices are refined and expanded upon.