# **Deterministic Text Extraction & Chunking — n8n Subtemplate**

This repository provides an N8N workflow for a **deterministic, code-based alternative** for text extraction and chunking. The workflow consists of two distinct routes, one for PDF documents and the other for Google Docs.

This workflow is not meant to be a substitute for more modern and adaptable forms of *agentic chunking*, as those have different strengths, but it is designed to work reliably on **free-tier environments** (e.g. N8N + Render), where the typical N8N nodes employed for agentic chunking may not work (specifically the Extract PDF Text node and the LangChain Code node).

In addition to providing a workaround in cases where agentic chunking may not be viable, This deterministic setup can also answer the need for a more reliable solution for text extraction and chunking, in all those cases where repeatable results are a key requirement.

This workflow is a partial fork of Cole Medin’s [*Ultimate n8n Agentic RAG Template*](https://github.com/coleam00/ottomator-agents/blob/c664279e48763ae37aa55a31dd1bdd41d29fcadf/n8n-agentic-rag-agent/Ultimate_Agentic_RAG_AI_Agent_Template.json), focusing only on the text extraction + chunking section.

## **What’s changed vs the original**

As mentioned above, the workflow consists of two distinct paths, one for PDF documents and the other for Google Docs.

✅ **PDF Processing**

* Replaces the original Extract PDF Text → LangChain Chunking pipeline with a **3-node Mistral API setup** (extract - URL generation - chunk), plus an additional node for metadata leballing.
* Produces better OCR in many cases and avoids long-running node failures.

✅ **Google Docs Chunking**

* Leaves the original Extract Text from Document node unchanged
* Swaps **LangChain agentic chunking** for a **JavaScript Code Node** implementing **deterministic chunking**:  
  + Preserves sentences, bullet lists, and avoids splitting words.
  + Adjustable MAX\_CHARS limit (e.g. 1200).
  + **Repeatable results** — same input = same output.

✅ **Fully Compatible with Downstream Nodes**

* Output is formatted to match the output of the original LangChain node. Most importantly, both outputs (for PDF and Google documents) are reformatted to use the same metadata labels, so that they can be ingested into the same node in the sequence.

## **Why use this version?**

Use this template if you:

* Experience **hanging/exhausted LangChain or Extract PDF nodes in N8N**. This can happen if you run N8N on a free tier plan, as well as if you host N8N on a free Render plan.
* Prefer **predictable chunking** rather than LM-driven chunking.
* Want **easier control** over how text is split.

## **How to use**

1. Import n8n-deterministic-chunking.json → *Import from file* in n8n.
2. Add credentials for **Mistral**, **Google Drive**, and **Postgres/PGVector** as needed.
3. Adjust MAX\_CHARS in the Code Node if desired.

## **Notes**

This subtemplate is designed primarily as a **reliable alternative to LangChain-dependent chunking workflows in restrictive environments**. If you’ve experienced:

* ❌ **Nodes stuck on spinning forever** (LangChain or Extract PDF Text)
* ❌ **Connection lost / resource exhaustion errors** on free-tier n8n or Render
* ❌ **Inconsistent chunking output from LLM-based chunkers**

…then this deterministic solution will suit your needs.