“**Preparing PDFs for RAG Workflows**”（为 RAG 工作流准备 PDF）是 **Ontologize** 团队提供的一门核心实战课程，主要教授如何在 Palantir Foundry 中通过 **Pipeline Builder** 处理 PDF 文档，使其满足**检索增强生成（Retrieval-Augmented Generation, RAG）**架构的要求 1。**RAG** 的核心理念是让大语言模型（LLM）能够访问其训练数据之外的特定知识库（如公司的 PDF 手册），从而生成更准确、有据可查的回答，并避免模型幻觉 2-4。
以下是根据来源对该教程所涵盖的端到端准备流程的详细解析：

1. 数据接入与文本提取 (Text Extraction)

处理流程始于将 PDF 文件上传为 Foundry 的 **Media Set**（媒体集）资源 5, 6。

- 提取方法选择：根据 PDF 的性质，Pipeline Builder 提供了不同的提取策略 7：
- **Raw Text**（原始文本）：适用于“机器可读”的 PDF（即可以直接在文中高亮选择文字的文档）7。
- **OCR**（光学字符识别）：适用于扫描件、照片或非直接可读的 PDF 文档 7, 8。
- **Layout Aware**（布局感知）：用于处理包含表格、图像等复杂布局的文档 7。
- 精准范围控制：用户可以指定提取的具体页码（例如从第 16 页开始，排除目录和索引），以确保后续处理的数据质量 9。

2. 数据清洗与字符串整合 (Joining Array)

提取出的文本最初通常以**数组（Array）**的形式存储，每个元素代表一页 10。

- **Join Array** 操作：为了防止语义在翻页时被截断（例如一个句子跨越了两页），教程建议使用 **Join Array** 将多页内容合并为一个大的字符串（**String**）11, 12。这样可以确保在后续的“分块”步骤中保持思路的连贯性 11。

3. 文本分块 (Chunking)

这是 RAG 工作流中最关键的预处理步骤之一。**分块（Chunking）**是指将长文档切分为更小、更易管理的片段 12, 13。

- 必要性：
- 模型限制：嵌入模型和 LLM 都有**上下文窗口（Context Window）**限制，无法一次处理整个巨型 PDF 12, 14。
- 检索精度：当用户提问时，系统应返回最相关的“片段”（如半页纸），而不是整本百科全书，以提高回答的针对性 14。
- 实战配置：教程使用 **Chunk String** 节点，将分块大小设置为 **512 tokens**，并根据段落、句子和单词的边界进行智能切分 13, 15。

4. 计算向量嵌入 (Compute Embeddings)

嵌入（**Embedding**）是将文本转换为高维空间的**向量（Vector）**数字序列，用于捕捉文本的语义 16, 17。

- 语义搜索原理：通过比较向量之间的距离（如余弦相似度），系统可以识别语义相近的概念。例如，“拿铁”和“咖啡”在空间中的距离会很近，而与“飞机”则很远 16。
- 模型对齐：用户需选择特定的预训练模型（如 text-embedding-ada-002）来执行推理 17, 18。
- 关键原则：必须同时保留原始文本（用于给 LLM 生成回答）和向量数据（用于检索），且后续应用层必须使用与生成阶段完全一致的嵌入模型，否则检索结果会失效 17-19。

5. 本体化与应用集成 (Ontology Integration)

最后一步是将处理后的数据发布到 **Ontology**（本体）层 20, 21。

- 对象建模：创建一个“Chunk（分块）”对象类型，将分块编号（Chunk Number）设为主键，并将嵌入字段配置为 **Vector** 类型 18, 21, 22。
- 下游赋能：一旦这些数据进入本体，它们就可以直接被 **AIP Logic** 或 **AIP Agent Studio** 调用，构建出能够根据 PDF 内容进行实时问答的智能体应用 19, 22。

总结

该标题代表了一套工业级的非结构化数据处理标准。它不仅教导如何“拆解”PDF，更强调了如何通过向量化和本体建模，将杂乱的文档转变为可被 AI “理解”并精准检索的企业知识资产 23, 24。