**Evaluation Criteria and Metrics**

To robustly compare the two PDF preprocessing pipelines, we define several evaluation criteria spanning **text accuracy, chunk quality, and downstream task performance**. Both quantitative metrics and qualitative assessments will be used:

1. **Text Fidelity and Completeness:** *Does the pipeline extract all the text correctly?* We will measure **token/character consistency** between the pipelines and against ground truth:
   * For PDFs that have a digital text layer (the majority of our test set), we can treat the text extracted by PyMuPDF (Pipeline B) as a near-ground truth (since it should exactly match the PDF content). We will calculate the total number of tokens and characters extracted by each pipeline per document and compute a **coverage ratio**. Ideally, a perfect extraction yields 100% of the original text. Pipeline A may score slightly lower if OCR misses some text or garbles characters. We will use **Levenshtein edit distance** or a similar text similarity measure between Pipeline A’s concatenated text and the PDF text to quantify OCR error rate (character error rate). Additionally, we will identify if any whole sections are missing. A token coverage significantly below 100% for Pipeline A on a digital PDF indicates some content was dropped (perhaps a figure caption the model didn’t detect or text OCR failed to recognize).
   * On scanned PDFs where no embedded text exists, we cannot do a direct fidelity comparison (there is no “ground truth” readily available unless we manually transcribe or have an alternative OCR). In those cases, completeness will be judged by manual checking: did the pipeline capture all paragraphs visible? Pipeline B’s coverage on scanned PDFs will be near 0% by nature, so the focus there is that Pipeline A is able to get at least a reasonable amount of text. We’ll flag any obvious missing chunks (e.g., if a page had two columns and the model only detected one consistently).
   * **Metric:** We will report **Character Recognition Accuracy** (CRA) for Pipeline A on digital PDFs = (1 – CER) where CER is character error rate vs. the actual PDF text. Also, **Text Coverage %** = (extracted characters / total characters in PDF text \* 100). Pipeline B on digital should have ~100% for both by design, serving as a sanity check. These metrics gauge the **fidelity** dimension.
2. **Chunk Coherence and Segmentation Quality:** *Are the chunks well-formed pieces of text (not too large, not cutting in the middle of ideas)?* We will evaluate chunking quality via:
   * **Sentence Boundary Alignment:** We will calculate what fraction of chunks end with a proper sentence termination (e.g. period, question mark, etc.) and begin with an uppercase letter or bullet, etc. A high percentage means chunks coincide with sentence boundaries, indicating coherence. Pipeline B is expected to nearly always end chunks at sentence boundaries (except rare extremely long sentences needing split). Pipeline A might have some chunks that end arbitrarily (if a paragraph ended mid-sentence due to a layout quirk or OCR dropping the last part). We’ll also check how often a single sentence is split across two chunks; this should be 0 for Pipeline B (by design) and hopefully low for Pipeline A if OCR did well. This gives a **Sentence Coherence Score** (percentage of chunks that are whole sentences or groups of sentences, not fragments).
   * **Paragraph Integrity:** We will see if paragraphs from the original document are kept intact in one chunk or unnecessarily split. For Pipeline A, this is usually true by its layout nature (one paragraph = one region chunk, unless the model splits it weirdly). For Pipeline B, if a paragraph exceeded the token limit, it will be split into multiple chunks. That’s not an error per se, but we will note the average paragraph split count. Using PDF text, we can heuristically detect paragraph breaks (e.g., double newline or indentation) and see if any pipeline ever merges two distinct paragraphs into one chunk (which would be a coherence issue). Pipeline B could merge paragraphs if the PDF text block spans them (some PDFs don’t mark a new paragraph clearly and just have a line break; our pipeline might treat it as one block and thus one section text – possibly chunked but still in one section).
   * **Chunk Size Uniformity:** Although not directly a “quality” in terms of comprehension, it affects usability in systems. We will calculate statistics on chunk length (in tokens) for both pipelines: mean, median, variance. Pipeline B should have fairly uniform chunk sizes centered near the max (500 tokens), except last chunks of sections may be shorter. Pipeline A might show a wider variance: some short chunks (titles, short paragraphs) and some very long chunks (if an entire page-length column was one region). A high variance or extremely large max chunk size in Pipeline A could be a downside for certain applications (e.g., a chunk of 2000 tokens might be too large to embed or use directly in an LLM prompt without further processing). We’ll document the largest chunk sizes encountered for each. Ideally, we want chunks that are not too large to handle. If needed, we might later apply a secondary split to Pipeline A’s output for fairness, but the evaluation will note if that is necessary.
   * **Human Coherence Rating:** We plan an optional **manual review** where a few sample chunks from each pipeline (for a given content) are shown to human evaluators (or domain experts). They will judge if each chunk “makes sense” on its own – does it read like a complete thought or is it confusing out of context? They will also check if the chunk boundaries seem logical (e.g., would they have combined or split differently?). Each chunk can be rated on a 5-point scale for coherence/integrity (5 = perfectly self-contained; 1 = very badly cut or mixed content). We expect Pipeline B to score high here, and Pipeline A to score well on most but perhaps lower on cases where OCR missed a line (chunk appears to end abruptly). We’ll average these ratings for each pipeline as a **Chunk Coherence Score (human)**.
3. **Structured Information Preservation:** *Does the pipeline capture document structure and elements that matter?* This criterion looks at things like:
   * **Section Detection Accuracy:** We will compare the section headings detected by each pipeline to an expected set of section titles in the papers. For instance, most academic papers have sections like Introduction, Methods, Results, Conclusion, etc. Pipeline A’s model classifies some text as "Title" which often catches the main title and possibly section titles (depending on how it was trained – PubLayNet’s "Title" might only mean the document title, not every section heading). Pipeline B uses font size; it might catch most top-level headings. We will manually list actual section headings from a few sample PDFs and see how many each pipeline correctly identified and placed. This gives a sense of **structural fidelity**. A pipeline that misses many section breaks might still have all text, but it loses the logical grouping which can be important for navigation.
   * **Tables and Figures:** If the documents have tables or figures, does the pipeline preserve their content? Pipeline A will give figure captions as text chunks (with label "Figure") and maybe some partial OCR of the figure itself (which might not be very useful). Pipeline B will provide the actual image file for figures and structured data for tables. We’re not directly evaluating table extraction accuracy here, but we note it as part of completeness – e.g., if a table’s text was completely missed by Pipeline A’s OCR or if Pipeline B’s find\_tables failed to detect a table, that’s missing content. We might measure this as a binary metric per table/figure: **table captured** or not, **figure image captured** or not, per pipeline. This contributes to the **completeness** dimension beyond just raw text.
   * **Metadata and References:** We’ll check if important metadata (title, authors) are captured. Pipeline A will likely OCR the title and authors as part of page 1 regions (hopefully labeled Title or Text). Pipeline B might capture title via metadata and also as a section heading on page 1. If a pipeline failed to clearly output the title or mixes it with other text, that’s a usability issue. Also, references section: Pipeline A might treat each reference as a separate region (perhaps as a list item if detected), whereas Pipeline B might lump the whole references section text and chunk it. Neither is “wrong,” but if one wanted to build a citation retrieval system, how the references are chunked could matter. We won’t heavily quantify this, but we’ll observe if references are present and complete in the output.
4. **Retrieval Performance (RAG Evaluation):** *How effective are the chunks for retrieving answers to questions?* This is a key evaluation from the downstream perspective. We will use an information retrieval test harness:
   * We will compile a set of **query questions** relevant to the content of the PDFs. For example, for each PDF, 2-3 questions that can be answered from it. Some questions might be factual (dates, numbers, definitions) and others conceptual (e.g., “What were the main conclusions about crop yield?”). If available, we could use the article’s abstract or conclusions to form questions, or have a domain expert pose questions.
   * Using each pipeline’s chunk outputs, we will index the chunks (likely using a vector embedding model suited for short texts). For each query, we retrieve the top *K* chunks that the system deems relevant. We then check if the chunk containing the correct answer (or information to derive it) is within the top K results. We will measure **Recall@K** (with K=3 or 5 typically) for each pipeline’s chunk set: e.g., “Recall@5 = 80%” means in 80% of queries, the correct answer’s chunk was among the top 5 retrieved. We will also compute the **Mean Reciprocal Rank (MRR)** of the answer-containing chunk, which accounts for ranking (a higher MRR means the relevant chunk tends to appear higher in the list, on average) – this is sensitive to whether the chunk that has the answer is usually the first result vs. sometimes third, etc.
   * This evaluation will highlight if one chunking method yields better retrievability. We expect Pipeline B’s more granular chunks to possibly improve recall (because specific information might be isolated in a chunk that matches the query closely, rather than being diluted in a larger chunk). However, if chunks are too small, sometimes a query might not have enough overlapping terms and could be missed. Pipeline A’s larger chunks might contain the answer plus other context, which might either help or hurt retrieval depending on the method (for vector similarity, extra unrelated text can add noise).
   * **Hallucination Rate in QA:** We will also run a small QA with an LLM (e.g., GPT-4 or a domain-specific model) where the LLM is given the retrieved chunks and asked to answer the question. We then evaluate the answers for accuracy. A **hallucination** in this context is when the model’s answer includes information not supported by the retrieved text. This can happen if the chunks are missing context or if the model tries to infer something beyond the given content. We will manually verify answers against the document. Hallucination rate can be computed as the percentage of answers that contain any incorrect or unsubstantiated content. We suspect that if a pipeline missed a piece of text (like the model’s OCR skipped a sentence), the LLM might fill that gap incorrectly, raising hallucination risk. Conversely, if chunks are well-structured and complete, the model can stay grounded. This is a more qualitative metric, but important for downstream quality: **Lower hallucination rates** are better for a reliable QA system. We’ll compare the two pipelines by counting hallucinations over a set of QA trials (likely a subset of the retrieval queries where we have a known answer to check against).
   * Additionally, we might measure the **answer accuracy** (how many questions were answered correctly when using each pipeline’s data). This depends on both retrieval and the model. It’s a high-level end-to-end measure: e.g., “Using Pipeline B chunks, the QA system answered 85% of questions correctly, whereas with Pipeline A chunks it answered 80% correctly.” This combines all factors (text accuracy, chunking, retrieval errors, etc.) into an ultimate impact metric.
5. **Efficiency and Resource Utilization:** *Which pipeline is more efficient for processing documents?* While quality is paramount, in practical deployment the speed and computational cost are also vital:
   * **Processing Time per Document:** We will record the time taken to process each PDF in the test set with each pipeline. This includes model loading time (for Pipeline A, loading the layout model) which we might amortize across documents, and the per-page processing. We will calculate the average time per page for each pipeline and the average (or total) time per document. If using a GPU for Pipeline A’s model, we’ll note that environment. We expect to find Pipeline B dramatically faster. For instance, prior benchmarks show PyMuPDF can extract text from pages a few times faster than even optimized C-based tools like Poppler​[documentation.help](https://documentation.help/pymupdf/app1.html" \l ":~:text=Again%2C%20%28Py,6%20times%20faster%20than%20xpdf" \t "_blank). In our test, if Pipeline B processes a 10-page article in, say, 2 seconds, and Pipeline A takes 30 seconds, that is a significant difference.
   * **Memory and CPU/GPU Usage:** We will qualitatively note resource usage. Pipeline A loads a deep CNN (ResNet-50 + FPN backbone from Detectron2) which can consume a few hundred MB of VRAM and CPU time for OCR. Pipeline B uses lightweight operations (mainly parsing, which is CPU-bound but not heavy, and spaCy NLP which is also CPU but can be moderate depending on model size – we used the small model, en\_core\_web\_sm, which is quite light). We might not instrument detailed memory profiling, but if any pipeline has issues processing larger documents (for example, Pipeline A might have trouble if a PDF has 100+ pages due to memory or simply long runtime), we will document that.
   * **Scalability:** As a metric, we could define how many pages per minute each pipeline can handle on a given hardware setup. For instance, *pages processed per second* as an approximate throughput measure. PyMuPDF has been reported to be able to parse text extremely fast (the documentation notes it’s multiple times faster than alternatives)​[documentation.help](https://documentation.help/pymupdf/app1.html" \l ":~:text=Again%2C%20%28Py,6%20times%20faster%20than%20xpdf" \t "_blank). We will measure on our hardware; e.g., Pipeline B might achieve >10 pages/sec on average, whereas Pipeline A might be around 0.2–0.5 pages/sec (if OCR is the bottleneck). These numbers will be gathered to inform the practicality of each approach for large-scale use (like processing thousands of documents).
   * We will present the timing results in a table, e.g., average seconds per page, total time for the entire test set for each pipeline, etc. This will highlight the trade-off: quality vs. speed.
6. **Overall Chunk Usability (Human Judgment):** Finally, beyond the technical metrics, we consider **user/end-user perspective**: if a developer or researcher were to use the JSON output for their NLP tasks, which would they prefer?
   * We will have a few annotators or colleagues subjectively compare outputs for a couple of documents. They will consider factors like: readability of JSON (structure clarity), ease of locating specific information in the JSON, and whether the chunking makes sense for their needs (say they were going to feed it into a QA system or browse the content).
   * For example, if one JSON clearly separates sections and even provides tables separately (Pipeline B), a user might find that more convenient than parsing text out of an OCR chunk that included a table in text form (Pipeline A). On the other hand, if Pipeline A’s output closely mirrors the document layout, a user might find it easier to map back to the original PDF when needed (because of the visual correspondence via bboxes and labeled figures).
   * We can collect a simple ranking: which output is preferred overall for usefulness. This is not a formal metric but will be noted in our report discussion. It ties together many of the above factors (accuracy, structure, completeness) into a single subjective evaluation.

Each of these criteria will be measured and compared between Pipeline A and Pipeline B on the same set of documents. For convenience, **Table 1** below summarizes the key metrics and evaluation methods:

| **Evaluation Aspect** | **Metrics / Checks** | **Details** |
| --- | --- | --- |
| **Text Fidelity** | Token/Character Coverage (%) | % of original text captured (by comparing to PDF text or reference) |
|  | Character Error Rate (CER) | OCR errors vs original text (Pipeline A mainly) |
| **Chunk Coherence** | Sentence Boundary Alignment (%) | % of chunks that end at sentence boundaries (no mid-sentence splits) |
|  | Paragraph Integrity | Check if paragraphs are intact or split/merged |
|  | Chunk Size Stats | Distribution of chunk lengths (mean, max, stdev in tokens) |
|  | Human Coherence Rating (1–5) | Average score by evaluators on chunk self-containedness |
| **Structure & Completeness** | Sections Detected (count, match %) | Compare identified headings to actual document sections |
|  | Tables Extracted (count) & Figures Extracted | Whether tables/figures content are present in output |
|  | Metadata Captured (Y/N) | Title/authors present correctly in output |
| **Retrieval Performance** | Recall@K (K=3,5) | % queries where correct chunk is in top K retrieved |
|  | MRR (Mean Reciprocal Rank) | Average 1/rank of the relevant chunk for queries |
|  | Answer Accuracy (%) | % of questions answered correctly using the chunks |
|  | Hallucination Rate (%) | % of answers with unsupported content (lower is better) |
| **Efficiency** | Processing time (sec/page) | Average time per page per pipeline​[documentation.help](https://documentation.help/pymupdf/app1.html" \l ":~:text=Again%2C%20%28Py,6%20times%20faster%20than%20xpdf" \t "_blank) |
|  | Throughput (pages per minute) | Derived from above, to indicate scalability |
|  | Resource usage (qualitative) | Notes on CPU/GPU and memory usage |
| **Overall Usability** | User preference / feedback | Subjective comparison of JSON readability and usefulness |

Table 1: Summary of evaluation criteria, metrics, and methods for comparing the PDF preprocessing pipelines.