DocAI PAI Best Practices

Table of Contents

Summary

The DocAI Incubator *Production Acceleration Initiative* (PAI) is a set of assets and Best Practices for DocumentAI project production implementations. This document outlines the general Best Practices topics and procedures that we recommend. Some aspects may be used for Proof of Concept (PoC) Projects.

NOTE: This is an early version of this document, so some content is in outline form and still delivered via consultation.

1. Use Case Analysis

Use Case Selection - What Doc Types, Business Unit, IT Resources.
Recommend to use the PAI Discovery Questionnaire (request this) to assess each use case, and gauge the difficulty.

1. Pilot(/PoC) - First use case: **Achievable** (moderate difficulty), with High Business Value
    a. PoC (Proof of Concept) - Effort to show DocAI will solve business need and will scale
    b. Pilot - Effort to roll a preliminary system into PRoduction

    c. PoT (Proof of Technology) - Effort to show that DocAI tech can solve business needs (not necessarily to scale)

2. Production - Most use cases: All except the most Difficult Use Cases
3. Difficult - Tough use cases: Highly Variable/Complex/High Accuracy. Recommendation is to **wait until the team is more experienced.**

The following domains have high difficulty:

- **Medical** - complex, specialized terminology, highly varied folders of docs (case mgmt)
- **Legal** - contracts are mostly narrative text, no layout signals. Long fields (Clauses and paragraphs very tough to extract)
- **Insurance** - esp. risk analysis
- **Financial** - esp complex tables
- **Handwritten** - OCR is Garbage In, Garbage Out (GIGO) - Very little can be done with poor images and poor handwriting (though, check for new OCR "prior knowledge" add-ins).

2. Design Targets

Identify and document the following:

Document Types

Prioritized Fields

Phased Target Accuracies

2.1 Prioritized Fields

We recommend categorizing fields by priority, typically into 3 tiers, such as:

Priority meaning	**Typical Names**	**(Google) Px Terminology**
High (Critical)	Critical / Mandatory	P0
Medium (Important)	Standard / Core	P1
Low (Nice to have)	Additional / Informative	P2

Best Practice is to **initially only focus on top priority fields**. Once the critical fields' accuracy is achieved, then medium field accuracy can be focused upon. Often non-critical field accuracy is improved by the work on critical fields.

Caution and phasing is indicated if customers have a large number (ex: 100) of fields they think are 'high priority' fields. While it is generally most efficient to make one pass through a dataset to label all at once. tackling all of those at once is risky (a recipe for disaster). A subset/Proof of Concept phase is indicated so that team skills and procedures can be tuned on a smaller effort.

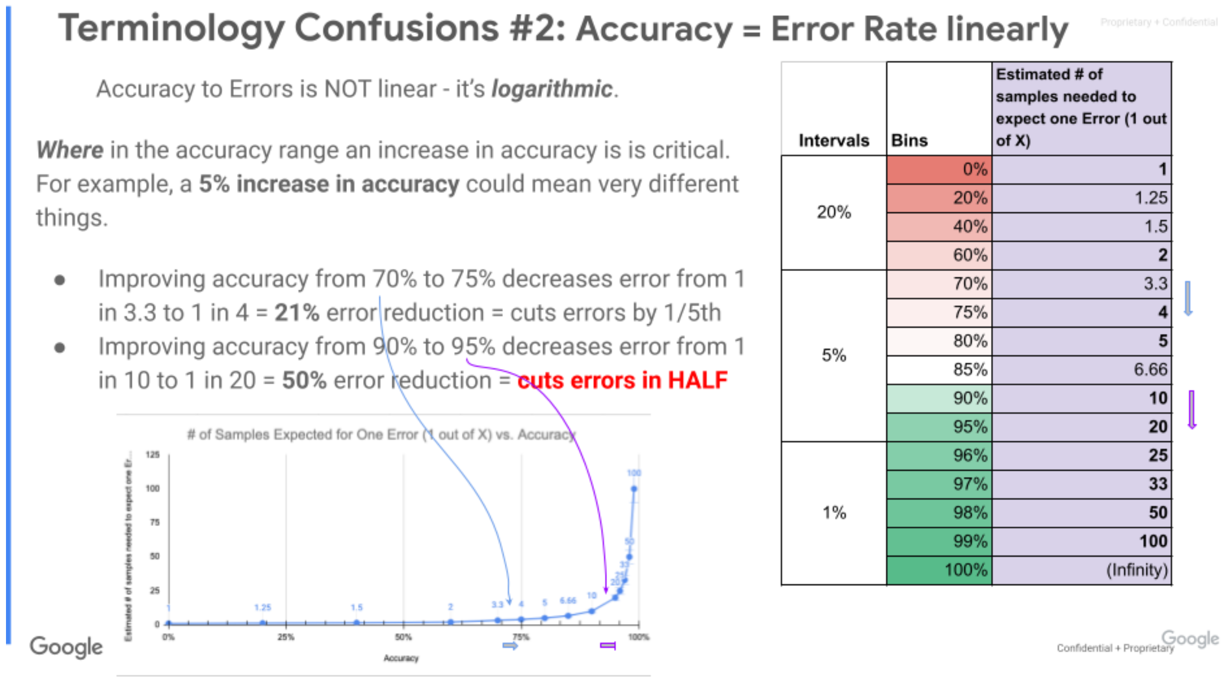
Some other factors that should be considered if phasing in fields:
- Number of fields to handle per iteration vs. number of passes to label documents
- Rank fields by complexity. It is recommended to start with both 'easier' and more difficult fields. Only doing easy fields or hard fields will affect expectations away from the overall picture.

2.2 Phased Target Accuracies

Every business process owner wants 100% accuracy ASAP. In most cases, 100% is not achievable and should not be set as the target. People are typically 97-98% accurate in human review. So a small error rate should be expected. The target accuracies for critical fields should reflect this.

Note: This document refers to “accuracy”. It means the **F1 metric**, which is the industry standard.

In practice, it takes time to get high accuracy, and the **difficulty becomes logarithmic** the higher above 90% accuracy.



Terminology Confusions #2: Accuracy = Error Rate linearly

Proprietary + Confidential

Accuracy to Errors is NOT linear - it's ***logarithmic***.

Where in the accuracy range an increase in accuracy is is critical. For example, a **5% increase in accuracy** could mean very different things.

- Improving accuracy from 70% to 75% decreases error from 1 in 3.3 to 1 in 4 = **21%** error reduction = cuts errors by 1/5th
- Improving accuracy from 90% to 95% decreases error from 1 in 10 to 1 in 20 = **50%** error reduction = **cuts errors in HALF**

Intervals	Bins	Estimated # of samples needed to expect one Error (1 out of X)
20%	0%	1
	20%	1.25
	40%	1.5
	60%	2
5%	70%	3.3
	75%	4
	80%	5
	85%	6.66
	90%	10
	95%	20
1%	96%	25
	97%	33
	98%	50
	99%	100
	100%	(Infinity)

# of Samples Expected for One Error (1 out of X) vs. Accuracy
Estimated # of samples needed to expect one Er...
125
100
75
50
25
0
0%
25%
50%
75%
100%
Accuracy
1
1.25
1.5
2
3.3
4
5
6.66
10
20
25
33
50
100
Google

Confidential + Proprietary Google

Therefore, it is useful to have lower prior targets, even for critical fields, to show progress, and to enable good business value while working on the final or highest target. So, a **phased approach to accuracy is recommended**, with the first target being the minimum viable production accuracy. This is a business decision, not strictly technical

For example, here is one set of phased accuracy goals that could be generally recommended:

Phase 1	Phase 2	Phase 3
85% (1 error out of 6.6)	**90%** (1 error out of 10)	**95%** (1 error out of 20)

Recall and Precision Metrics

Some people may be interested in Recall and Precision metrics. There is always a tradeoff between these, and basically, the F1 score indicates the best combination of the two.

In practice, there are very few use cases where these metrics matter for document extraction.

- High Precision - means that the entities that are extracted are highly confident, and increases the risk of undetected entities (false negatives). For example, a business need might be to make sure the *SSN* is the correct text (if extracted) is critical to get right.

- High Recall - means that all possible entities are extracted, at the risk of including wrong entities (false positives). For example, a business need might be to get all possible instances of a field like *signers* or *contacts.*

3. Model/Labeling Work

3.1 Annotation Quality

3.1.1 Labeling Instructions

Thorough labeling instructions need to be developed, illustrating each field and common variations and questions.

3.1.2 Labeling Spin Up

It is recommended that **small batches (<10) be labeled initially, and those reviewed very carefully** for quality, prior to ramping up labeling volume. It is common to take several cycles until high quality is achieved. Unless this is done, it is likely that many documents will need to be re-labeled.

3.1.3 Quality Control

To achieve the highest accuracy possible with [up]training it is essential to ensure **every document in the dataset has been thoroughly reviewed** for any mistakes, such as:

- Mislabeled entities,

    - Ex: net_amount labeled as total_amount
- Entities not labeled,
    - All instances of a label should be labeled. Any unlabeled instances of the same text will send an anti-pattern to the model. Ex: supplier_name is labeled only for the logo, but occurs elsewhere on the document.
- Bounding boxes capturing text that should not be part of an entity…
    - Labeling must be highly consistent for the model to be consistent. Ex: money values should typically only be THE NUMBER and not include the currency symbol.
- Empty labels (unless supported)
    - Until supported, no empty labels should be present, and may cause an error.

    - Request the **Empty Bounding Box Removal** code tool from your Google contact.

The reviewer(s) should be THE experts in what each field is (per the business). A workflow that **escalates** questions about labeling to intermediate and top experts may be recommended for large annotation jobs.

3.1.4 No Text Changes for Training Purposes

For [re/up]training purposes ZERO manual direct text changes should be put into the dataset. The only changes should be by changing what OCR text is selected, either with Bounding Box or Text Cursor select. If the correct text string was selected, but the OCR is wrong , that is OK for training (not for Production). For production purposes, the value may be mis-recognized by OCR. The value SHOULD be corrected for downstream entry into the customer system of records.

So intake of new documents into the model's dataset that parsed poorly need to be reviewed in order to filter out the OCR corrections, (and any other labeling problems). Only BBox changes should be added to the dataset. Incubator Tooling can help this detection of WHAT was changed.

3.2 Golden TestSet - By Processor

A "Golden TestSet" of documents is a **primary Best Practice** in Machine Learning and DocAI.

> *Test Dataset: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. The Test dataset provides the gold standard used to evaluate the model. … The test set is generally well curated. It contains carefully sampled data that spans the various classes that the model would face, when used in the real world.*
>
> Link

(The DocAI TEST dataset is called the "Validation" dataset in this link.)
(Other relevant definitions: Wiki, StackExchange (2nd comment)).

In DocAI, the Golden TestSet's purpose is to **represent the documents that will be encountered in production** for accuracy measurement purposes. This should consist of document types & subtypes to be predicted upon by a processor by occurrence rate in Production.

These documents **must NOT exist in the processor's TRAIN dataset.** It must be independent, in that the model must not have "seen" these documents before. The **TEST dataset is a great place** for the Golden TestSet, since it will be used after training for evaluation and you will get the F1 Scores.

The F1 results of running a parser on the Golden TestSet should be **THE accuracy metric** of a parser. It is the **yardstick** used to measure the effectiveness (accuracy) of a DocAI process over time. When a process achieves a target accuracy on the Golden TestSet, it should be declared that it has achieved that milestone.

Since it must be labeled with Ground Truth, the labeling must be very carefully reviewed for accuracy. It is best that this be very carefully curated and **stable** (no changes). Carefully building a comprehensive Golden Test Set early, by being kept stable, establishes a reliable apples-to-apples metric to evaluate all experiments and possible approaches for high accuracy.

If differences are found in a given processor's accuracy (F1) between the Golden TestSet and production documents, then the Golden TestSet does not accurately reflect production documents, and should be changed if the difference is significant. Note, your TestSet is now a different comparison tool than in the past, and past scores may not correlate.

NOTE: While TEST and TRAIN dataset should start with the same distribution of (layouts of) documents (reflecting PROD distribution), the best practice is to **add more samples for tougher to parse variations to the TRAIN dataset**. So the two datasets will typically not end up with the same distribution.

3.2.1 Alternative Test Sets & Evaluation Procedures

You may desire (and it is generally a good idea) to evaluate on various other sets of documents, so you can assess the accuracy of your parser on various subsets or clusters of documents, including by language, supplier, customer, domain, layout source or region.

There are two methods to do evaluation on alternative test sets:

1) Import the Processor to another processor, and populate that test set to do evaluations in the UI
2) Use the Evaluation API on a GCS folder of documents, to obtain the evaluation data via API

3.3 Training Datasets

3.3.1 Primary Recommended Approach: Curated Tranches

To build your training dataset, we recommend

A) to start with a limited **random** sample of production documents. then,
B) the best practice is to add to the TRAIN dataset with documents **specifically selected** (curated) for tough (poorly performing) fields and clusters (styles/layouts) of documents.

The problem with adding **random** documents to a dataset is that those random documents added are likely to include doc styles that are already performing well. The model is unlikely to change much as your training is not focused on its weak points. So to overcome this, we recommend being much more selective of the documents used in the CDE **training** process or for uptrainable processors, the **uptraining** process. That is, “curate” the addition of documents.

3.3.2 Minimum Dataset Size

The size of the starting random dataset recommended is generally a few hundred, depending on a number of factors, including:

- availability of production docs
- the difficulty and cost of labeling
- the degree of variability in the documents

In MOST (but maybe not all) production use cases, minimal dataset sizes of 40 or 50 documents will not yield a significant improvement in uptraining, nor usable accuracy in CDE. But if the documents are highly consistent some accuracy could be achieved. Initial training in tranches of 50 or 100 are recommended.

3.3.3 Large Datasets

If you already have a large (> 5000 samples) **randomly** selected dataset (either unlabeled or labeled), esp. if you are not getting your desired accuracy, we recommend **restarting** with randomly selected 500 to 1000 documents from your dataset and uptrain or retrain a new model.

There are major advantages to keeping the total dataset size limited.

1) Large datasets cost more to make
2) Large datasets (over 10k) tend to be more difficult to improve. A larger dataset makes the model more “static” and not as easy to change. It will take more samples of a field or doc style to get an improvement.

   For example, the first 1k documents will impact the model performance and behavior a lot more than the next 1k documents, and the impact of adding 1k docs when there are 20K docs will be much less than the first 1k groups.

Just released as preview in Aug 2023 is the CDE **Selective Labeling** feature, which will select the best 30 documents to label from an unlabelled dataset (using a “diversity filter”). This can be used (repeatedly) to reduce a dataset size. If docs are already labeled, then mapping unlabelled docs to labeled docs will be needed.

3.3.4 Adding a Subclass to a Dataset (Ex. a Customer or Region)

Often a business process would like to add a certain kind or source of documents to a parser, such as a new customer or region (a subclass). The default approach to this is often to just add X number of sample documents. X is often in the 50 to 100 range, dependent on the size of the dataset, larger if the dataset is larger.

This approach makes general sense if the subclass has a highly consistent layout, such as from one business. There are two aspects to consider, both in the interest of minimizing the number of documents that need to be added to the dataset for a given level of accuracy:

- If the subclass doesn’t have variety - is **very consistent,** as from one business, then a **few documents should be tested** to see the accuracy. If the accuracy is relatively high, then the training can start with fewer documents. If accuracy is low, then start with a higher number of documents. This would be the only “curation” of highly consistent subclass.
- If the subclass **has variety,** has a number of layouts, then the addition of documents should be curated first. The first (and subsequent) tranches of documents should FIRST be run through the parser and **only the poorly parsed documents should be added**. That is, EVERY document should be run through the parser, and then those documents must be *curated*, filtered to **only add the poorly parsed ones**.

The main point is that adding documents that **already parse well** does not help the accuracy of the parser and bloats the dataset size. See above section 3.3.3 on the impact of the size of the dataset.

3.4 Uptraining Guidance

3.4.1 Uptraining vs CDE Considerations

In general, if an Uptrainable pre-trained parser exists and is relevant, it should be the first option considered. Uptrainable pre-trained parsers generally have advantages of needing fewer (if any) dataset samples to achieve good accuracy. However, if

- 1) the pre-trained schema is **not substantially the same as desired schema**,
  Ex: critical fields are not covered, or Parent/Child field groups need to added
- 2) if the document type fields do not well align with the use case corpus,
  Ex: service orders use case may have service date and hours line-item content,

then it is worth an experiment to see if direct use of CDE has advantages.

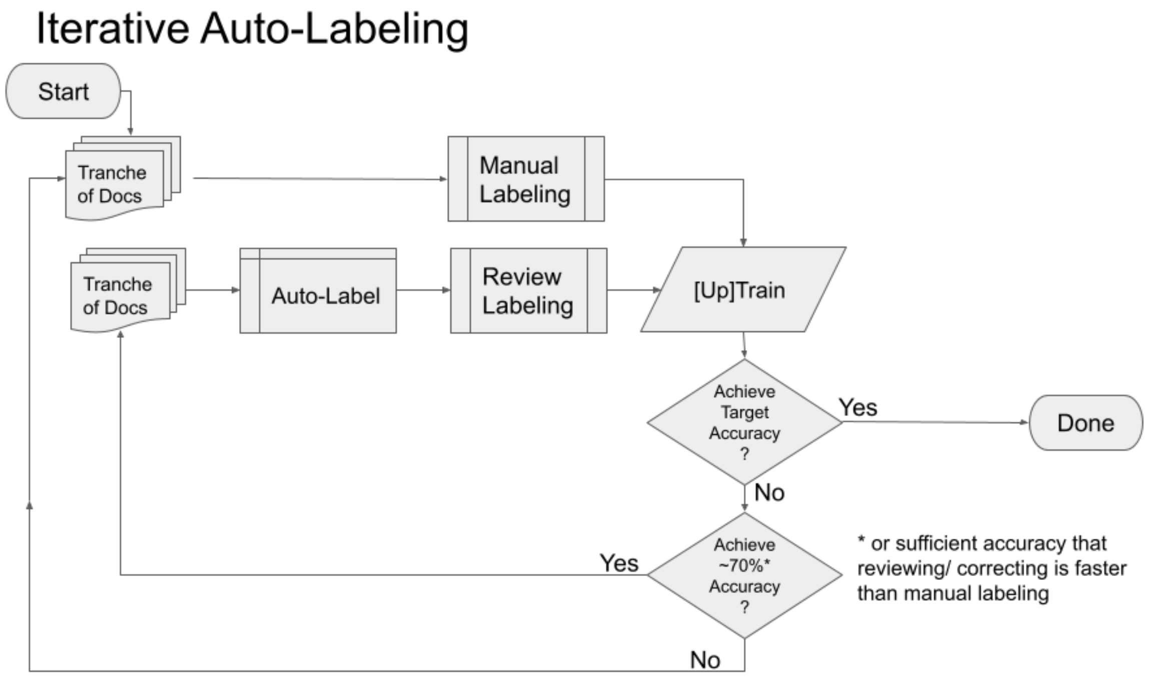
3.4.2 Base Processor Selection

There may be multiple uptrainable parsers to consider. For example, with a Bill of Lading use case, both Invoice and Purchase Order processors should be considered and tested. There may also be different versions of the processor. When uptraining, we suggest to first try your test set with different versions of the base processor (if there are different versions).

Different versions can yield different accuracy results. Usually the latest version will give the best results, but that may not always be the case. So we recommend trying uptraining on both the RC version(s) of the Invoice and Expense and the latest stable versions of the processors on the **exact same dataset** (for apples-to-apples comparison). This can be **easily done** in the DocAI console with a few clicks. An eval with the Golden TestSet is extra evaluation data.

3.5 Iterative Auto-Labeling

To keep the manual labor of labeling to a minimum, a Best Practice approach is to use the Auto-Labeling feature of DocAI iteratively. Start with a small set, [up]train, **then when the accuracy is good enough that correction is faster than full manual labeling**, **then use auto-labeling** to reduce labeling workload, review/correct and iterate on this process.



Tranches of 50 to 100 are generally recommended.

Generally, when the accuracy is at least 70% or so, significant labor can be saved by correction instead of complete manual labeling. However, this depends on the nature of the corrections being made. If there are many parent child grouping errors, correction can be more time-consuming.

3.6 Adding Fields (CDE and UpTraining)

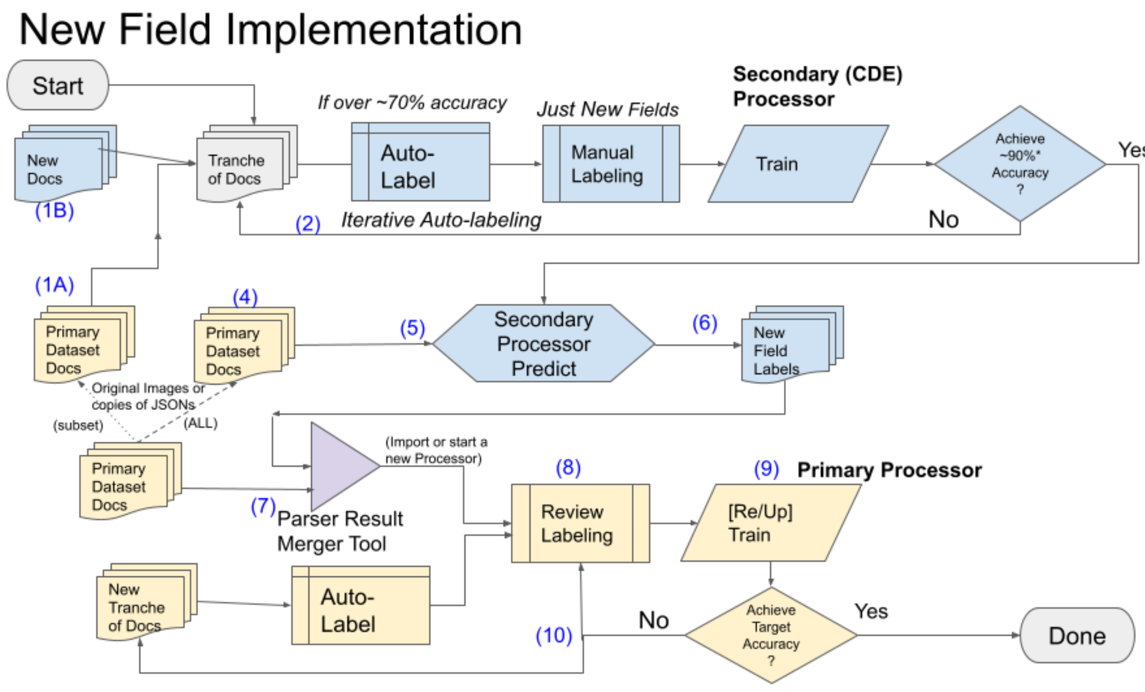
There are situations where one or more additional fields need to be added to a Processor's schema. While it is possible to manually label every document in a processor's dataset, this is often a heinous annotation workload, if

A) the number of documents in the dataset is high, and/or

B) if additional documents need to be added to the dataset (because the new fields are rare in the existing dataset) in order to have enough to train on.

Any added documents have to have ALL fields annotated, though auto-labeling can minimize this workload.

There are ways to minimize the workload of adding new fields, described here, as a best practice. We refer to the processor where the fields need to be added as the "primary" processor.



Fundamentally, the basic approach relies upon **training a second ("secondary") CDE processor** on just the fields to be added. This is a temporary measure to ease the annotation workload.

The documents used to train this model should either be (step 1A in diagram) in the primary processors dataset or (1B) be new documents VERY like them, and of course should be documents with the desired new fields. We recommend using (2) Iterative Auto-Labeling (section 3.4) to build this processor and (3) train to a high level of accuracy (like 90%). The exact target level of accuracy depends on the difficulty of the field(s), the difficulty of labeling them and the cost of review for the error rate.

Once the secondary CDE model is accurate enough, it's prediction could be run on the documents in the primary dataset to label (essentially, auto-label) them. So (4) prepare either

- original images or
- copies of the JSON files

and (5) predict (process) them with the secondary processor.

The only problem is prediction on the document JSON files REPLACES existing labels (entities). Whether with original image files or JSON, these (6) secondary processor result files only have the new fields.

Therefore, (7)
request the Incubator **Parser Result Merger** code tool to INSERT the new field entities into the existing dataset JSON files. Import or start a new processor. Then review the labels, and up/re-train. Use standard methods to improve accuracy if needed (10). Either review labels, or add additional samples.

In summary the procedure looks like this:

A. **Build the secondary CDE** with the new fields to a high level of accuracy
B. **Export** the primary process dataset
C. **Copy** the exported JSON's; keep one set of JSONs for import to a new processor
D. **Run** the (copied) JSONs (or original image files, if the correlation of JSON to image is known) through the secondary processor to get secondary processor output JSON files
E. Use the Incubator **Parser Result Merger** tool to copy the new field entities from the secondary processor output JSON files to the JSONs for import to a new processor
F. **Import** the combined JSONs into a [new or the primary] CDE or Uptrained processor.
G. The documents will have the new fields labeled, but they **must be reviewed**.

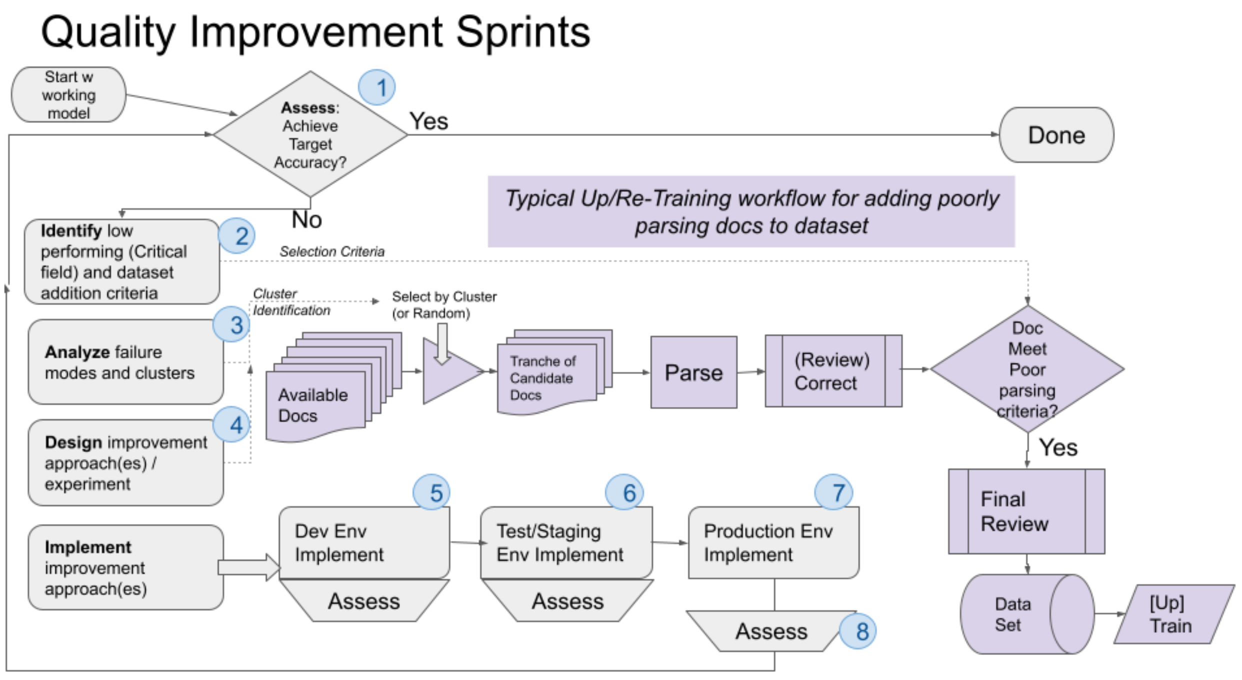
4. Quality Improvement Sprints

This section discusses the recommended cycle of improving model-based accuracy (for Workbench and uptrainable processors).

Standardized report format/content recommended on each quality improvement sprint is recommended. This informs all stakeholders of efforts, targets and timelines, to help set expectations.

4.1 Overall procedure

Repeat these steps for each sprint (modeling approach steps shown):


Quality Improvement Sprints
Start w working model
Assess: Achieve Target Accuracy?
1
Yes
Done
No
Typical Up/Re-Training workflow for adding poorly parsing docs to dataset
Identify low performing (Critical field) and dataset addition criteria
2
Selection Criteria
Cluster Identification
Select by Cluster (or Random)
Analyze failure modes and clusters
3
Available Docs
Tranche of Candidate Docs
Parse
(Review) Correct
Doc Meet Poor parsing criteria?
Yes
Design improvement approach(es) / experiment
4
Final Review
Implement improvement approach(es)
Dev Env Implement
5
Assess
Test/Staging Env Implement
6
Assess
Production Env Implement
7
Assess
8
Data Set
[Up] Train

Steps Overview:

1. **Assess** Accuracy rate of critical fields going in,
2. **Identify** Low performing fields and docs, compare to target accuracy
3. **Analyze** the failure modes and “cluster” of those,
4. **Design** Improvement Approaches to fix issues. This often involves **experimentation** and **adding docs to the dataset.**
5. **Implement** Approaches and **Assess** in Dev Env
6. **Implement** Approaches and **Assess** in Test/Staging Env
7. **Implement** Approaches in Production Env
8. **Assess** accuracy in Production Env

4.2 Procedure Steps in Detail

1. Start with an Assessment of Accuracy rates of critical fields going in. This can be the results of the Eval of the last training. Best if the assessment is against the Golden TestSet.
2. Identify Critical Low performing fields, by comparing to current phase to target accuracy
3. Analyze the failure modes and “cluster” of those,
   Failure modes include:
   a. Field Not Found
   b. Wrong Text Selected
   c. Partial FIeld Found

   d. OCR errors [very limited approaches to fix, leave out of parser iterations]
   e. Post-Processing logic errors
   f. Other failure Modes

   Cluster dimensions include:
   - Customer / Supplier
   - Layout
   - Region
   - Language

4. **Design / (experiment with)** Improvement Approaches to fix issues
   a. Modeling: To add curated (select) samples that parse poorly to dataset,
   b. Rule: Add or modify logic to post-processing / workflow
   c. (OCR: Select an Alternative OCR Engine in DocAI)
5. Implement Approaches in Dev / Staging
   a. Modeling: Identify samples, [auto-]label, Add samples to dataset, [Up]Train
   b. Rule: Add logic to post-processing / workflow
   c. OCR: Alternative OCR Engine

6. Test / Assess accuracy with Tests
7. Implement Approaches in Production
8. Test / Assess accuracy in Production

4.3 Discussion

Here is a more narrative description of the cycle:

1. Start with a set of F1 scores for your parser

2. Determine the **weak critical fields** that need strengthening for each quality improvement sprint cycle. Determine your criteria for the sprint:
   A) an error on a weak field, or
   B) N number of errors, indicating a poor parse, an outlier

3. Identify (best via some cluster) a tranche (50-100) of candidate documents.

   Identifying candidates: Segmented approach:
   This could be a good time to use any segmentation of your documents that you have. If you have documents segmented (classified) via offboard metadata (prior knowledge) by **document Sub-type** (Hotel receipts, car rental receipts…) or any **other classification or clustering metadata** (info you already have about documents) criteria, including:
   - Customer / Supplier

- Region
- Language
- Layout (typically a manual selection)

Random Approach
Else just run a variety of documents through

4. Parse the Candidate Documents.
Run the candidate documents through your model.

5. Correct (Review) or compare to GT labeling
If you have Ground Truth (labeled docs), keep a copy of these, and you can use the below tooling. Else use HITL to correct and use Incubator tooling (**Pre-Post Hitl Parser And Ocr Issue Identifier**) to detect errors of the just-parsed docs. Else manually review the parsed documents and correct any errors.

6. Use tooling to select poorly parsed documents

   Documents that perform poorly per your criteria you should add to your next uptraining iteration. Incubator tooling ((**Identifying Poor Performing Documents**)(tool) can assist this, performing the selection programmatically from folders of pre- and post-HITL JSONs.

   For example, to select all documents with less than some level of accuracy, either by
   - document (N number of errors) or
   - by an error on certain critical fields.

   The same logic can be inserted into production workflows.

7. Perform a final review of the resulting tranche of those poorly performing documents

8. Add the resulting tranche of those poorly performing documents (already labeled) to your (Dev) dataset. As the dataset grows you may need to add more documents per cycle. If the dataset gets large, the resulting tranche size needed could increase to say 100 documents, then 200 documents, depending on the accuracy increase that you see for a certain size of tranche.

9. [Up]Train and evaluate results to ensure an improvement.

10. Roll out to Staging and Production environments and assess accuracy.

11. If resulting accuracy in Production is still not satisfactory, repeat these steps with another quality improvement sprint.

4.4 Expectation Management

Users should not expect much changes in processor performance from one or even a few added documents with a certain pattern of correction. Depending on the size of the dataset and how much of an outlier from the main patterns the failing layout is, it is hard to determine the number of samples to improve the performance on the outlier pattern.

Add in small tranches, measure accuracy against your golden dataset or a test set for that outlier pattern (if significant), and repeat until the accuracy target is achieved.

4.5 Quality Improvement Sprint Standardized Report

Standardized report format/content recommended on each quality improvement sprint is recommended. This informs all stakeholders (esp. business) of efforts, targets and timelines, to help set expectations.

Here is suggested content of such a report:

Sprint ID
Sprint Timeframe
Sprint Scope:
- Use Case
- DocType
  - Critical Fields being addressed
  - Current Accuracy, Target Accuracy

Sprint Stage: Status/ETA:
- Discovery
  - Determining Failure modes and “clustering”
- Design/ Development
  - Determining /Designing Approach
  - Executing Improvement
    i. Modeling: Identify samples, label, Add samples to dataset,
    ii. Rule: Add logic to post-processing / workflow
    iii. OCR: Alternative OCR Engine
- Testing
  - Development (unit) Testing
  - Quality Assurance Testing/Evaluation
- Deployment
  - Production Implementation
  - Production Accuracy (Post-) Assessment

5. Human Review Bypass Best Practices

There are best practices with regard to **selecting the criteria by which a workflow decision is made to bypass human review** (that is, “touchless” processing). The determination of bypass criteria is fundamentally a BUSINESS decision, not technical, and should be based on firm data: accuracy and error statistics.

The fundamental questions about bypassing human review are:

1) what would be the **error rate** of documents bypassing human review at a given confidence score threshold
2) what is the **tolerance of the business** to the above error rate.

5.1 Document Routing Product Feature

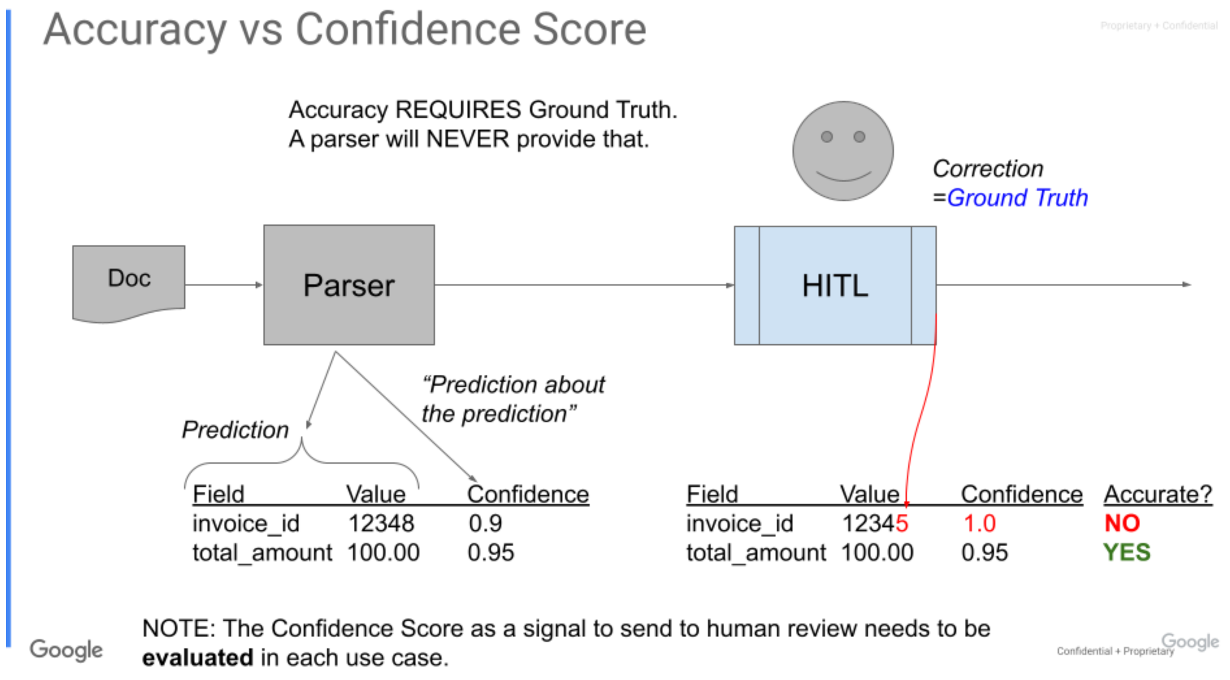
DocAI provides a no code criteria for routing documents to human review or to bypass our existing HITL, based on the **Confidence Score (CS)** of any field (document level filter) or selected fields (label-level filter).

- Doc level Filter: If the CS of **any field** is below the configured threshold, the document will be routed to HITL.
- Label-level filter: Each field/label MAY have a threshold set. IF the CS for the field is below the threshold, it will be routed to HITL.

If post-processing code is required, then this product feature cannot be used. But Incubator has the code to emulate this criteria and routing decision.

Ask for the **Custom HITL Review for JSON** code tool

NOTE: **Confidence Score is NOT accuracy.** Do not confuse the two. DocAI parsers always give CS with every prediction. You can consider the CS to be **a prediction about the extracted field prediction**. It tells you whether the parser “thinks” it is a firm or questionable prediction. But a **parser cannot determine if any prediction is accurate or not,** since that requires Ground Truth (GT). GT is typically supplied by Human Review, so only after HITL can accuracy be measured. Any correction is an error.



Accuracy vs Confidence Score

Proprietary + Confidential

Accuracy REQUIRES Ground Truth.
A parser will NEVER provide that.

Field	Value	Confidence
invoice_id	12348	0.9
total_amount	100.00	0.95

Field	Value	Confidence	Accurate?
invoice_id	12345	1.0	**NO**
total_amount	100.00	0.95	**YES**

Google

NOTE: The Confidence Score as a signal to send to human review needs to be **evaluated** in each use case.

Confidential + Proprietary Google

5.2 CS and Error Rate Data Collection

Best practice is for the determination of bypass criteria to be **data-based** (on a statistically significant sample size). To collect this data:

1) Initially **do not bypass any documents**
2) Capture all errors and CS's
3) Compile the data to determine the error rate at various CS thresholds

For example, in a simple use case if the total_amount field is being evaluated as criteria to bypass, then say on 100 documents you collected the above data, you would compile the data into the below table:

CS Bucket	Number of total_amount Errors (and percent in error)	Number of documents in CS Bucket
90-100%	2	10
80-90%	5	30
70-80%	10	20

Then this historical data indicates the following anticipated error rates at these different CS Thresholds:

CS Threshold	Anticipated % in Error if bypassed	Bypassed Volume (includes higher CS threshold volumes)
90%	2%	10%
80%	5%	40%
70%	10%	60%

5.3 Bypass Decisioning Best Practice

As stated above, the decision on what thresholds to set for bypass is a BUSINESS decision, based on the tolerance of the business process to error versus the benefit of reduced labor. Humans are generally regarded as having an 97-98% accuracy for document review work. So manual processes generally can tolerate a low error rate. While 100% accuracy is always the goal, in practicality, some errors should always be anticipated. For very error-intolerant processes, even triple-human review may not catch all errors, and are VERY expensive.

So based on statistical data as shown above, the business can make a cost/benefit analysis of the anticipated errors vs the savings of human labor to review. In the above example, perhaps a 5% error rate is acceptable, then 40% of the volume can be bypassed by setting the CS threshold to 80%. If higher accuracy is required, but 2% errors is acceptable, then setting the CS threshold to 90% will bypass 10% of volume.

5.4 Continual Auditing Recommendation

Document mixes change, and models "drift". So the common best practice is to still perform Human review ("audit") on a small (5-10%) of otherwise bypassed documents. Review of the error rate of these documents will keep track of the actual error rate of documents being bypassed.

The bypass criteria should be re-evaluated on a periodic basis, like a quarterly cadence. As the parser improves over time, often the criteria is raised because fewer errors will go thru.

5.5 Document Split Processing

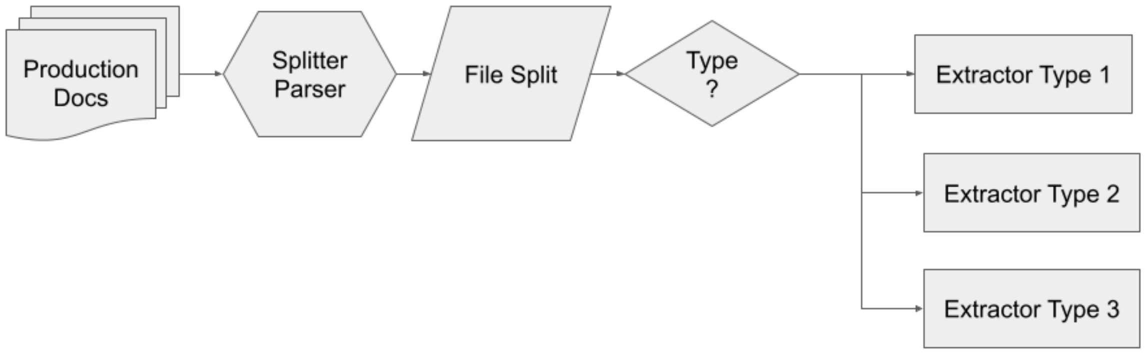
5.5.1 Split Review Recommendation

Designers may be tempted to directly use the (without human review) the predictions from a splitter. This sort of workflow is NOT recommended.

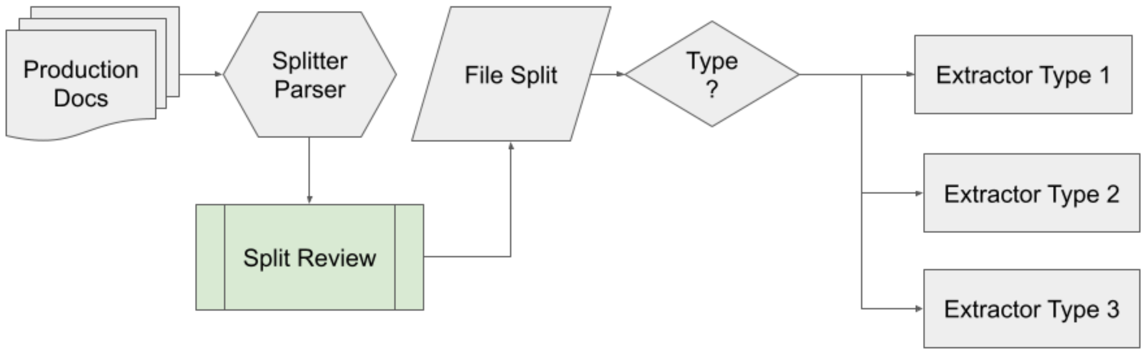
The prediction results of CDS (Custom Document Splitter) and any pretrained splitter **should almost always be reviewed by a human before the physical file split is performed** in the workflow. The reason for this is that if the split is wrong (and there will be errors, even with a highly accurate splitter),

- all downstream extraction tasks will be wrong (having the wrong pages in each doc), and
- it is typically very difficult to correct (to correctly reconstruct the documents) .

For example, an incorrect split causes two documents to be wrong. The extraction parsers will have the wrong pages to extract from. Getting those two poor documents out of the workflow, and the correctly split versions of them is a function that is complicated to implement.



This split workflow is NOT recommended



This split workflow IS recommended.

In summary, the cost of a split error is high. The cost of human review is relatively low.

5.5.1 Split Workflow Implementation

Splits may currently be reviewed using “Project Munich” - an open code/SDK web app framework. The future HITL on Document Warehouse product will also support Split Review.

Request the Project Munich codebase from your Google contact.

In summary, the cost of a split error is high. The cost of human review is relatively low.

- The code to physically split PDF's is found in these locations: https://cloud.google.com/document-ai/docs/splitters#:~:text=The%20following%20code%20sample%20uses%20Document%20AI%20Toolbox%20to%20split%20a%20PDF%20file%20using%20the%20page%20boundaries%20from%20a%20processed%20Document.
- https://cloud.google.com/document-ai/docs/handle-response#splitting:~:text=The%20following%20code%20sample%20uses%20Document%20AI%20Toolbox%20to%20split%20a%20PDF%20file%20using%20the%20page%20boundaries%20from%20a%20processed%20Document.
- https://cloud.google.com/document-ai/docs/handle-response#pdf-split
- https://cloud.google.com/document-ai/docs/toolbox#pdf-split

The future **HITL on Document Warehouse** product will also support Split Review.

6. Poor Prediction Problems

6.1 Measure Accuracy Rate

Anecdotal (single document) prediction problems do not necessarily indicate a general parser problem. The *first* step should be to **get an accuracy rate on a statistically significant number of documents** (50 or 100).

Any single prediction error could be expected within a given error rate. For example a 95% accuracy rate still has 1 error out of 20, Just because similar documents had different prediction results is not necessarily a problem.

6.2 Pre-trained processors

For pretrained processors that cannot be uptrained, there is nothing that can be done directly. If the measured accuracy rate is significantly below 90%, a bug may be filed so that Engineering is aware of a performance problem. Some samples would be highly recommended.

CDE or post-processing custom code may be used to get better accuracy. Sometimes CDE can be (quickly) constructed as a “gap-fill” to address the pre-trained parser weaknesses. This will incur a second API call charge per document, which many customers will not tolerate.

CDE can almost always be trained to achieve pre-trained parser performance on all fields but that often takes a significant number of samples in the training dataset. This annotation work can be done as part of production workflow to lower the total overhead. That is, simply route a number of pre-trained-predicted (but fully reviewed) documents to be added to the CDE dataset. This can be done in an iterative manner as noted in the Quality Improvement Sprints section.

6.3 CDE or Uptrained Label Quality

The basic guidance for CDE or Uptrained accuracy issues is to

A) Add documents that parse poorly to the training dataset and re-[up]train. If the documents to be added are carefully selected (curated), then the size of the dataset should stay relatively small, and added documents will have an impact.

B) Review labels for accuracy. Any inaccurate labels can impact your model's accuracy. All instances of a field should be labeled.

    a) Wrong text is labeled. If there is any question about whether the right text is labeled for a field, then a subject matter expert should be consulted or review the labels. The labels should reflect what the business considers the field.
    b) Not ALL instances of a field are labeled. EVERY instance should be labeled on every document, else the model gets an anti-pattern indicating that sometimes the field should not be labeled.
    c) Overlapped labels are not recommended. Any text on documents should only be labeled once. if a field needs to be broken apart, either label separately, or post-process to achieve that.
    d) Currently multi-line fields (blocks) are not well supported in CDE. Multiple single line fields are recommended.
    e) Bounding Box too large
    f) Bounding box overlaps
    g) Too many labels where the OCR text has been overridden (useless for training)
    h) Inconsistent labeling
        i) currency amounts with and without the currency symbol included
        ii) PO numbers with and without PO prefixes (and similar fields)
        iii) Company names with and without suffixes (LLC, Inc. etc)

6.4 "Key" Field Technique

One technique to improve recognition of a weak field, esp. when the parser confuses it with another field, is to add a field that is the "key" to the weak field.

Key-Value pairs are often found in forms. For example:

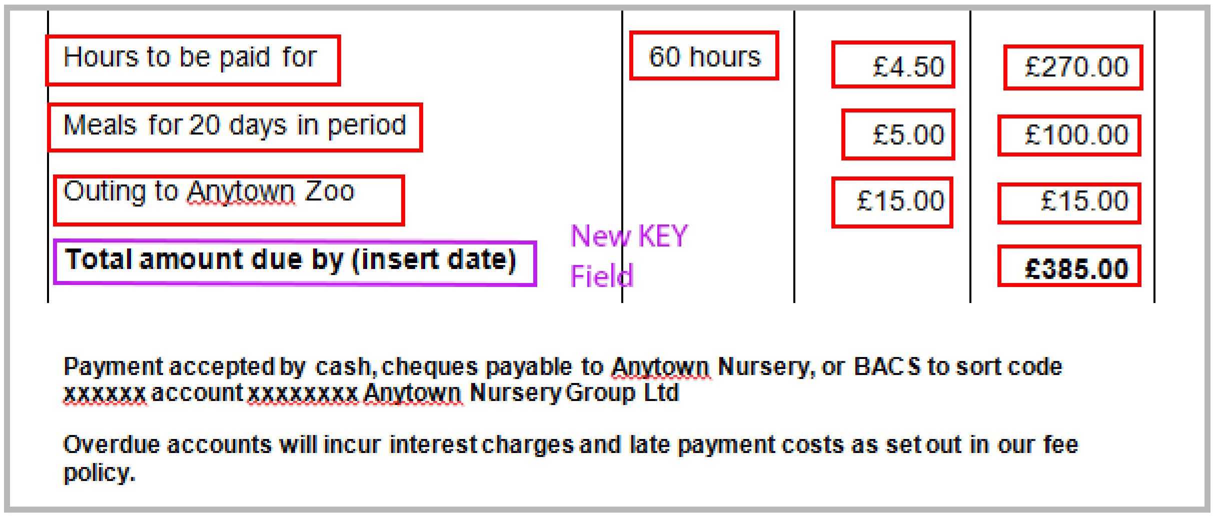
Name: Alfred Smith
Account Number: 12345

In this example, "Name:" and "Account Number:" are the keys, and the values are: Alfred Smith and 12345.

If you have a weak field, for example, **total_amount**, you should consider an experiment to **label the key text** for that field. In this case there are probably a number of different key text, like,

Total Amount:
Total:
Total Amt:

For example:


New KEY Field

Hours to be paid for	60 hours	£4.50	£270.00
Meals for 20 days in period		£5.00	£100.00
Outing to Anytown Zoo		£15.00	£15.00
Total amount due by (insert date)			**£385.00**

Payment accepted by cash, cheques payable to Anytown Nursery, or BACS to sort code xxxxxx account xxxxxxxx Anytown Nursery Group Ltd

Overdue accounts will incur interest charges and late payment costs as set out in our fee policy.

An experiment is recommended because there is significant work to label every document. The experiment would be to label a subset of documents with all such instances with a new field, perhaps called total_amount_**key**, and see if it increases accuracy and to what degree. IF successful, then the entire dataset can be labeled.

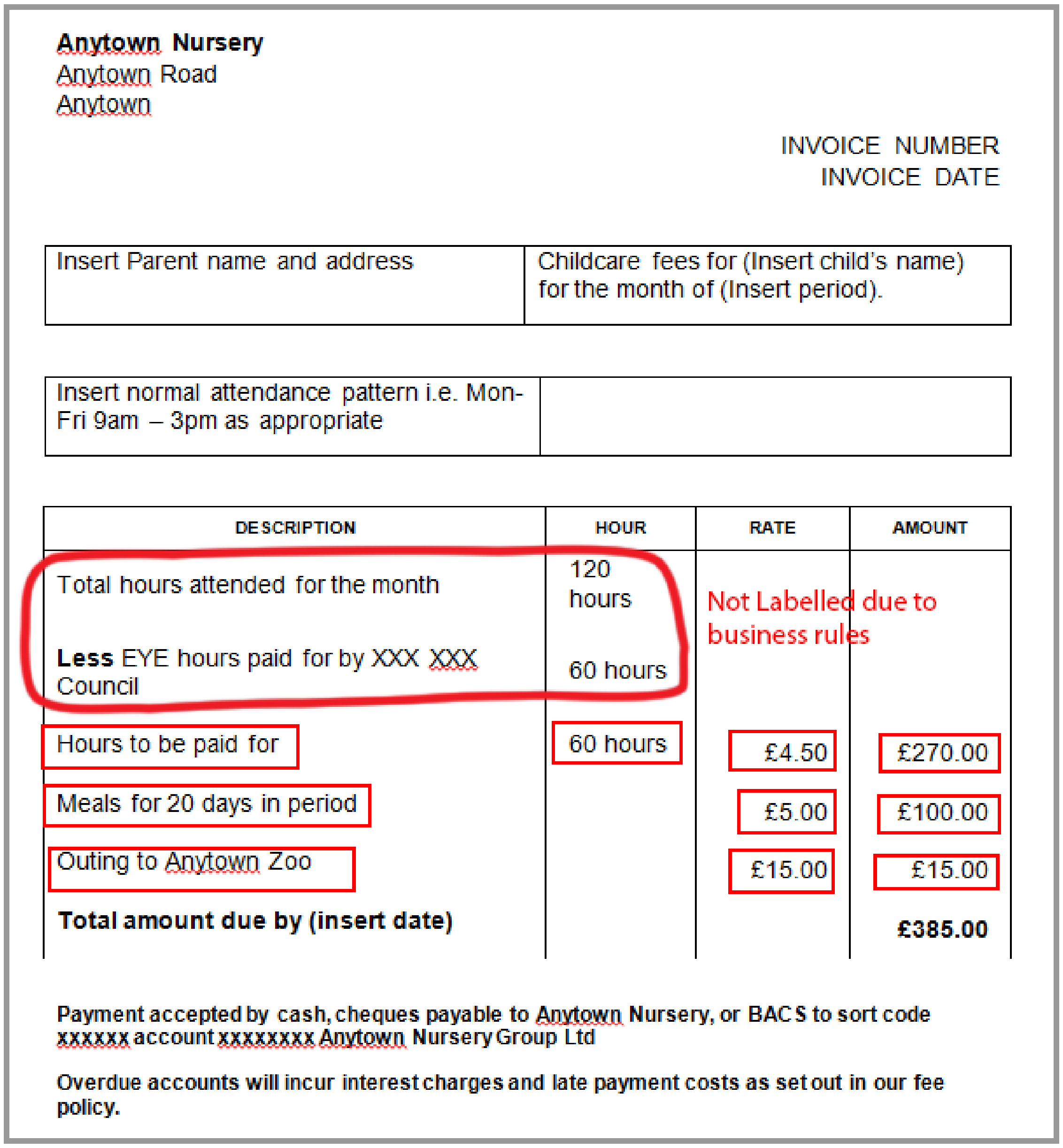
Incubator has a code tool to automatically label based on a synonym word list. This can significantly reduce the manual labeling work. Reminder, all new labels should be reviewed to ensure quality.

6.5 Consistency Principle

6.5.1 Consistent Labeling

There are some situations where the business needs may cause some fields to be labeled and NOT labeled in a manner where a parser may have difficulty following the pattern.

For example, business needs for invoicing may ignore certain partial or "Credit" line items. If this leads to the practice of **not labeling those line items, which creates a tough pattern** for a parser to follow. The signal for whether or not to extract the line item is relatively subtle: a positive or negative value or a value in one column or the absence of another column.



Anytown Nursery
Anytown Road
Anytown

INVOICE NUMBER
INVOICE DATE

Insert Parent name and address	Childcare fees for (Insert child's name) for the month of (Insert period).

Insert normal attendance pattern i.e. Mon-Fri 9am – 3pm as appropriate	

DESCRIPTION	HOUR	RATE	AMOUNT
Total hours attended for the month	120 hours		
Less EYE hours paid for by XXX XXX Council	60 hours		
Hours to be paid for	60 hours	£4.50	£270.00
Meals for 20 days in period		£5.00	£100.00
Outing to Anytown Zoo		£15.00	£15.00
Total amount due by (insert date)			**£385.00**

Not Labelled due to business rules

Payment accepted by cash, cheques payable to Anytown Nursery, or BACS to sort code xxxxxx account xxxxxxxx Anytown Nursery Group Ltd

Overdue accounts will incur interest charges and late payment costs as set out in our fee policy.

Whenever there is a clear pattern of text where some are labeled, but others are not, based on some business rule, that business rule may be difficult for a parser to be trained to recognize. If all the text was labeled consistently, that is easy pattern for a parser. **The instances that are not labeled may become an "anti-pattern" to the parser**, confusing it as to whether the text should be labeled or not, resulting in lower accuracies.

In these situations, the recommended best practice is to **label all such text consistently**. In our example, all line items should be labeled. Then the parser has a consistent easy pattern to follow. The recommendation is to then **enforce the business rule in post-processing**. In our example, It is easy logic to remove incomplete line items or credits from the entities, prior to downstream processing into the system of record.

6.5.2 Schema for Consistent Labeling

Another situation related to consistency (often found in invoices, for example) regards whether a field/label occurs once in a header or body, AND also occurs in a table, such as for line items. The recommendation is to make separate labels in these situations

For example, in some documents there may be a date (like service date) associated with each line item in an invoice. While the meaning of this field in the line items is very similar to an Invoice date in the header (that occurs once), it both has

- a bit different meaning (a date not for the whole document, but for a line item), AND
- it has a very different place and occurrence in the document.

In such cases, because they different patterns, two different labels are recommended,

- the standard **invoice_date** field, and
- a **lineitem_date** field

If one label is used for both kinds of data, the parser may have lower accuracy as it tries to recognize two different patterns of occurrence for the same field.

6.6 Other Approaches to Accuracy Improvement

If these approaches are not effective, then consider the following:

- the dataset, if large, should be curated.
- Post-processing approaches explored.
- Sometimes ensembling with Form Parser or another parser is indicated.

7. Diagnostic Best Practices (Troubleshooting)

Diagnosing DocAI issues has some best practices:

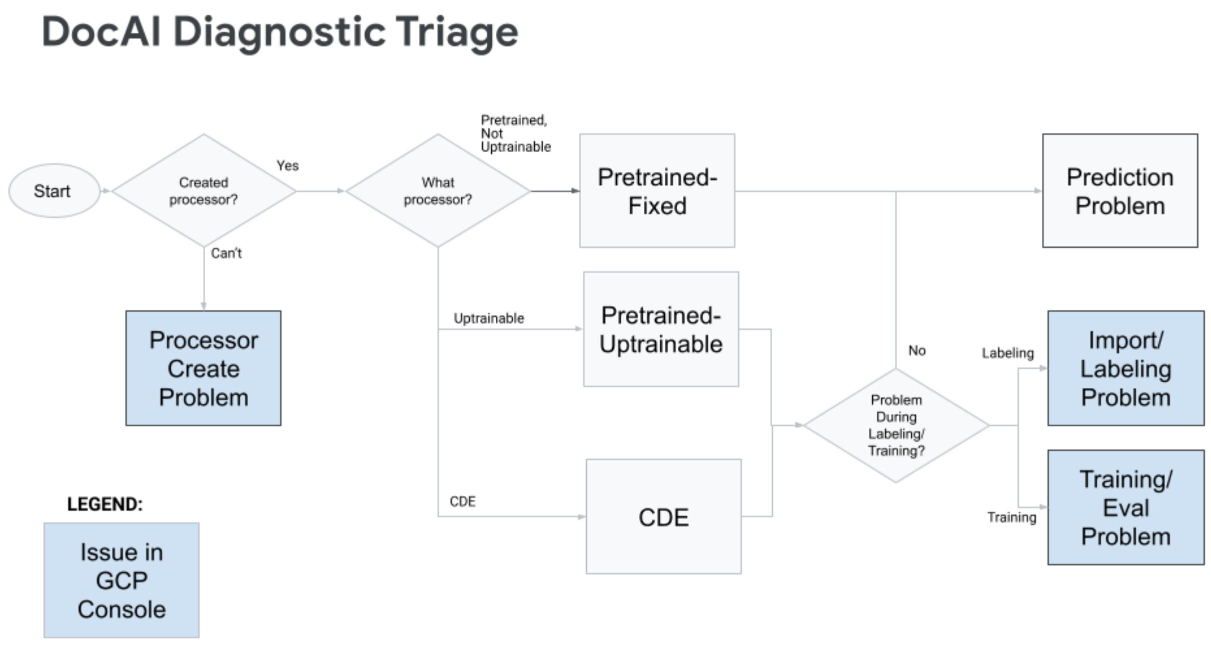
[WIP: Currently OUTLINE format]

7.1 General Approach: **Characterize** the problem

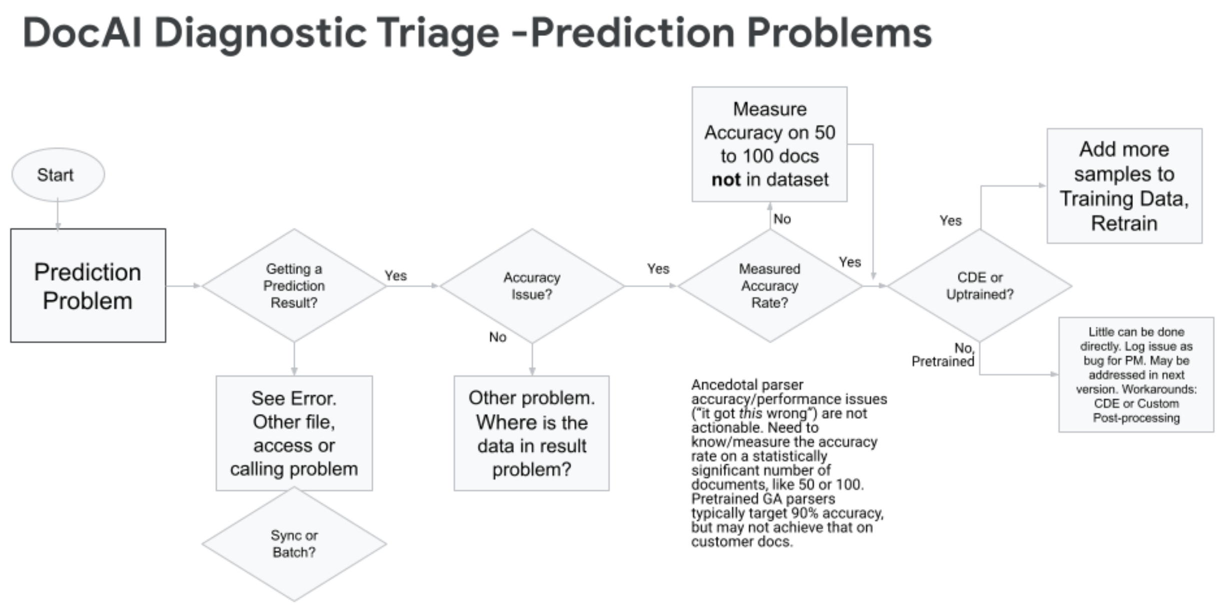
1) Characterize the problem,
2) aim for **a reproducible case** that Engineering can diagnose if not solvable by customer, etc.

- One document or many?
- Consistent or Intermittent?
  - Intermittent: (Get Occurrence rate and pattern)
- Is the problem
  - A PROCESSOR CREATE problem?
  - A IMPORT or LABELING problem
    - USABILITY PROBLEM,
    - an ERROR or
  - A TRAINING/EVAL problem

A (POOR) PREDICTION PROBLEM



7.2 Prediction Problems


DocAI Diagnostic Triage -Prediction Problems
Start
Prediction Problem
Getting a Prediction Result?
Yes
See Error. Other file, access or calling problem
Sync or Batch?
Accuracy Issue?
Yes
No
Other problem. Where is the data in result problem?
Measured Accuracy Rate?
No
Measure Accuracy on 50 to 100 docs **not** in dataset
Yes
CDE or Uptrained?
Yes
Add more samples to Training Data, Retrain
No, Pretrained

Little can be done directly. Log issue as bug for PM. May be addressed in next version. Workarounds: CDE or Custom Post-processing

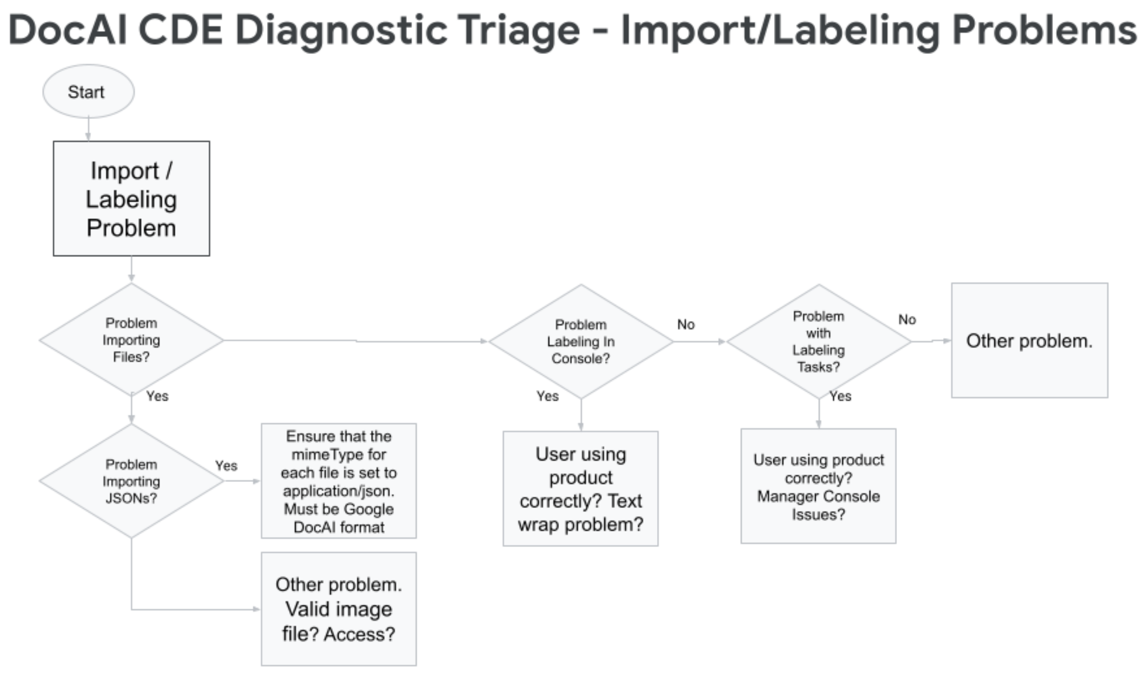
Ancedotal parser accuracy/performance issues ("it got *this* wrong") are not actionable. Need to know/measure the accuracy rate on a statistically significant number of documents, like 50 or 100. Pretrained GA parsers typically target 90% accuracy, but may not achieve that on customer docs.

Error problem (no result)?
Accuracy (poor parsing) Problem? (see section 6)

What is the measured accuracy on 50+ docs?
Is the processor CDE or Uptrainable?
Then review dataset labels, OR
Add additional samples

7.2 Import / Labeling Problems

CDE and Uptrainable processors:


DocAI CDE Diagnostic Triage - Import/Labeling Problems
Start
Import / Labeling Problem
Problem Importing Files?
Yes
Problem Importing JSONs?
Yes
Ensure that the mimeType for each file is set to application/json. Must be Google DocAI format
Other problem. Valid image file? Access?
Problem Labeling In Console?
No
Yes
User using product correctly? Text wrap problem?
Problem with Labeling Tasks?
No
Yes
User using product correctly? Manager Console Issues?
Other problem.

Import Problem?
Check mimeType in GCS
Check JSON format (must be Google doc.proto)
Check valid image / format
Labeling problem?
Usability problem?

7.3 Usability Problems

USABILITY PROBLEM - Difficulty using the DocAI product
What UI?
Is the behavior as intended (Typical) or is it anomalous? Are there examples of it behaving differently? Does the problem only happen with one document or multiple?
Is an additional feature being requested (a Feature Request)? or is something broken (not working as intended/advertised)?

7.4 Training Problems

Issues around training happen after training is triggered. There are two main categories:

- **Dataset validation errors**
  These are errors that occur at the beginning of training, as the documents that make up the TRAIN and TEST datasets are checked. Certain conditions may cause some labels and documents to be rejected. You may receive **document validation warnings** about certain labels not having a textAnchor. These are where during labeling the text from OCR was changed. Thus, the entered text does NOT occur on the document, and so no TextAnchor exists. These labels cannot be used for training. Rather than stop training, the label simply does not count toward the minimums required.

  If, after filtering these out, there are insufficient instance counts for a label, then a **dataset validation error** will be issued, and training will be aborted.

- **Quota errors**
  There are two quota related errors during training. These are
  - *DeployedCustomModelsPerProjectPerRegion* : The maximum number of deployed processors per project has been used for this project. Undeploy 1 or more deployed processors and train again.
  - *ConcurrentCustomModelTrainingPipelines* : The maximum number of concurrent training jobs in use for this project. Wait for the completion of current training jobs and try again.

- **Errors during training**
  In normal situations, after validation checks, no further errors should be expected during training. These should be reported, ticketed and sent to Engineering for diagnosis.
  Issues include:
  - Internal Error (13)
  - Training does not end within ~12 hours

7.5 Error Problems

A first effort should strive to determine if:

1. the error is induced by how an API is called (incorrectly), or unexpected input to an UI, versus
2. an error in proper use (a product bug).

what processor? What action or API call?
Is the error intermittent (Get Occurrence rate and pattern) or consistent?
Does the problem only happen with one document or multiple?
Has the actual call to DocAI API been examined (not via the client library)? Does that work outside of the workflow?

Is the error a 500 or 13 Internal Error (which should never happen)? or is it a 4xx error that indicates a problem with a request (usually via API)? Or does the error say, “Try again“ (usually via UI)?
CONTEXT:
Does the customer have any security measures? VCP-SC? CMEK? other?