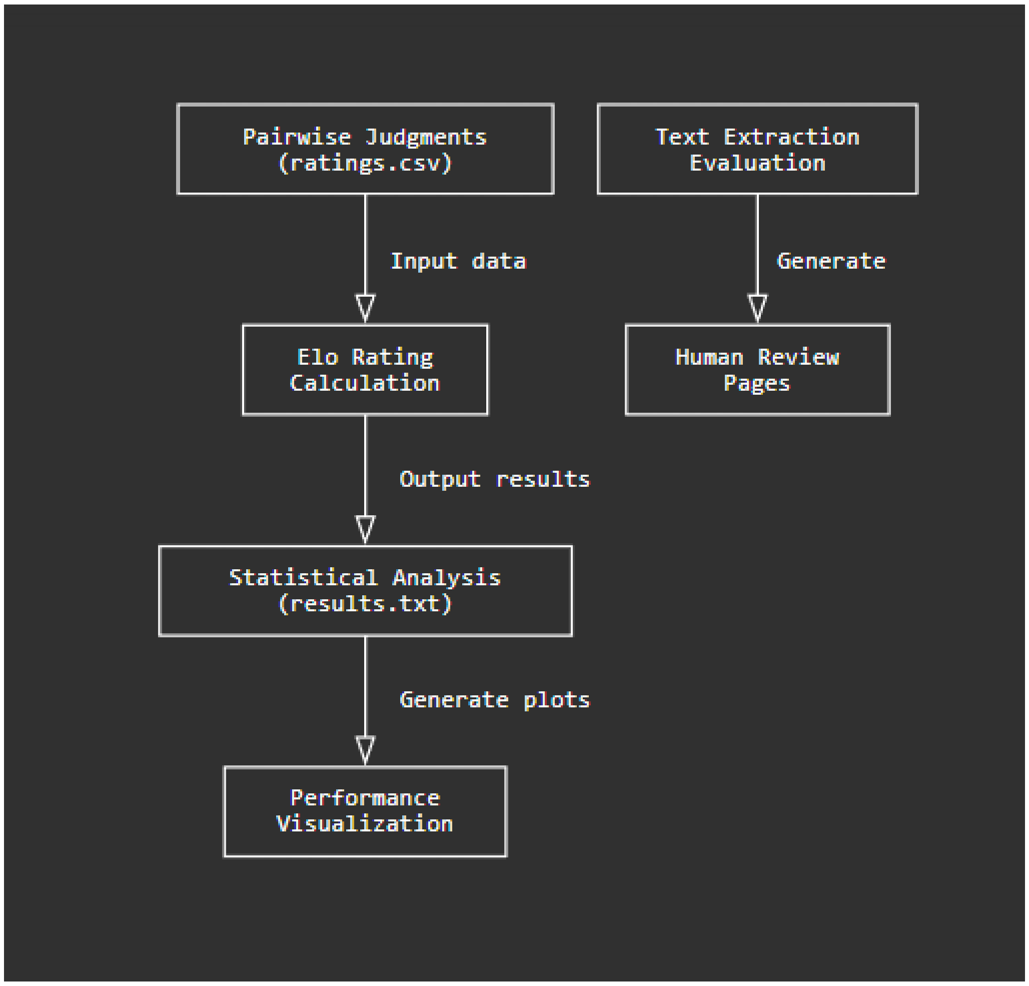

Pairwise Judgments (ratings.csv)
Text Extraction Evaluation
Input data
Generate
Elo Rating Calculation
Human Review Pages
Output results
Statistical Analysis (results.txt)
Generate plots
Performance Visualization

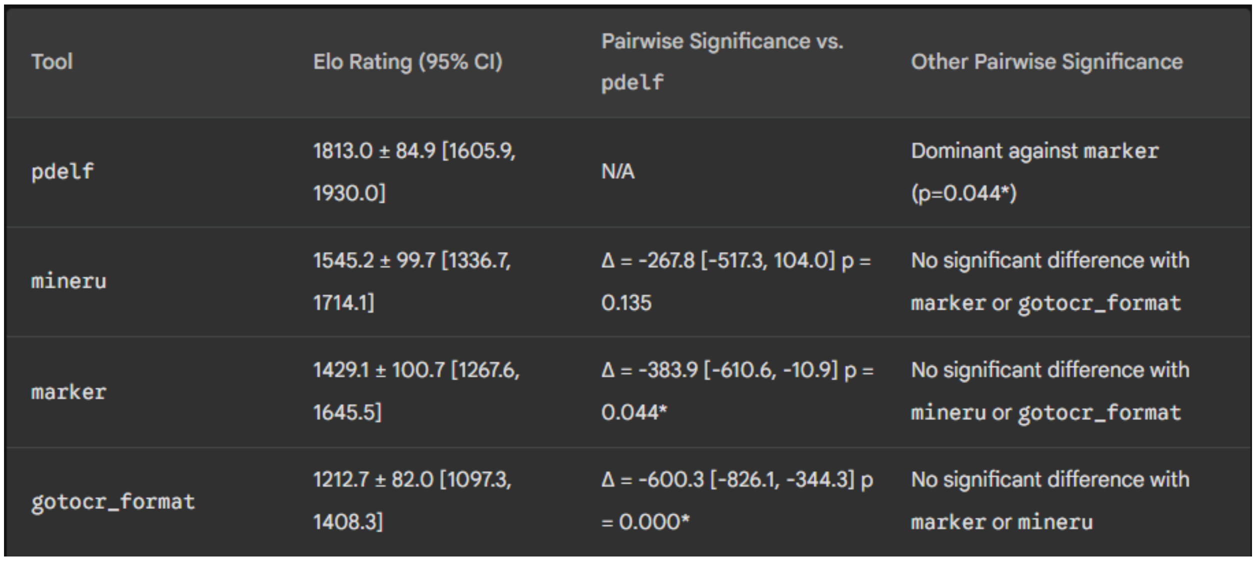
Benchmarking OCR tools and analyzing their performance involves a multi-faceted approach within the olmOCR project, encompassing objective evaluation frameworks and visualization tools. This includes calculating Elo ratings to compare OCR tools based on pairwise judgments, evaluating text extraction from PDFs using advanced similarity metrics, and generating various plots to visualize performance trade-offs and evolution.

The system utilizes an Elo rating framework to compare OCR tools, including olmOCR (referred to as pdelf in the data), against competitors. This process involves gathering pairwise judgment data, where different OCR methods are compared. Scripts in the scripts/elo directory process this data to compute bootstrapped Elo ratings with confidence intervals and perform pairwise significance tests. This statistical analysis identifies performance differences that are robust, rather than merely coincidental. The outcomes are then visualized using boxplots, providing a clear graphical representation of each tool's rating distribution and relative standing.

For evaluating text extraction from PDFs, the framework provides tools to compare different extraction methods and assess their accuracy. This involves calculating text alignment scores using sophisticated algorithms like Hirschberg and Needleman-Wunsch, which quantify the similarity between extracted text and a "gold standard." These evaluations can also incorporate human feedback. The system generates interactive HTML review pages, as seen in **scripts/eval/evalhtml_template.html**, for human evaluators to compare text extraction outputs. These pages allow users to vote on which text block is "better" or if other conditions apply, and this feedback is then analyzed to generate aggregated reports and ELO ratings for the various OCR methods. Further details on how these prompts are constructed and outputs viewed can be found in Prompt Generation and Document Viewing.

To visualize OCR model performance, the project generates various plots. A Pareto frontier plot, created by **scripts/plots/pareto_plot.py**, illustrates the trade-offs between OCR model performance and cost, allowing for the identification of models that offer the best performance for a given cost. Additionally, timeline plots, generated by **scripts/plots/plot_olmocr2_timeline.py**, depict the evolution of OCR model performance over time, showing how different versions of **olmOCR**, **Marker**, **MinerU**, and other models have progressed in their **Overall** scores.

ELO Rating System for OCR Tool Comparison



Tool	Elo Rating (95% CI)	Pairwise Significance vs. `pdelf`	Other Pairwise Significance
`pdelf`	1813.0 ± 84.9 [1605.9, 1930.0]	N/A	Dominant against `marker` (p=0.044*)
`mineru`	1545.2 ± 99.7 [1336.7, 1714.1]	Δ = -267.8 [-517.3, 104.0] p = 0.135	No significant difference with `marker` or `gotocr_format`
`marker`	1429.1 ± 100.7 [1267.6, 1645.5]	Δ = -383.9 [-610.6, -10.9] p = 0.044*	No significant difference with `mineru` or `gotocr_format`
`gotocr_format`	1212.7 ± 82.0 [1097.3, 1408.3]	Δ = -600.3 [-826.1, -344.3] p = 0.000*	No significant difference with `marker` or `mineru`

The **olmocr** project employs an Elo rating system to objectively compare the performance of various Optical Character Recognition (OCR) tools, including **pdelf** (which represents **olmOCR**). This system moves beyond subjective evaluations by using pairwise judgment data to calculate relative strengths of OCR tools, similar to how chess player rankings are determined.

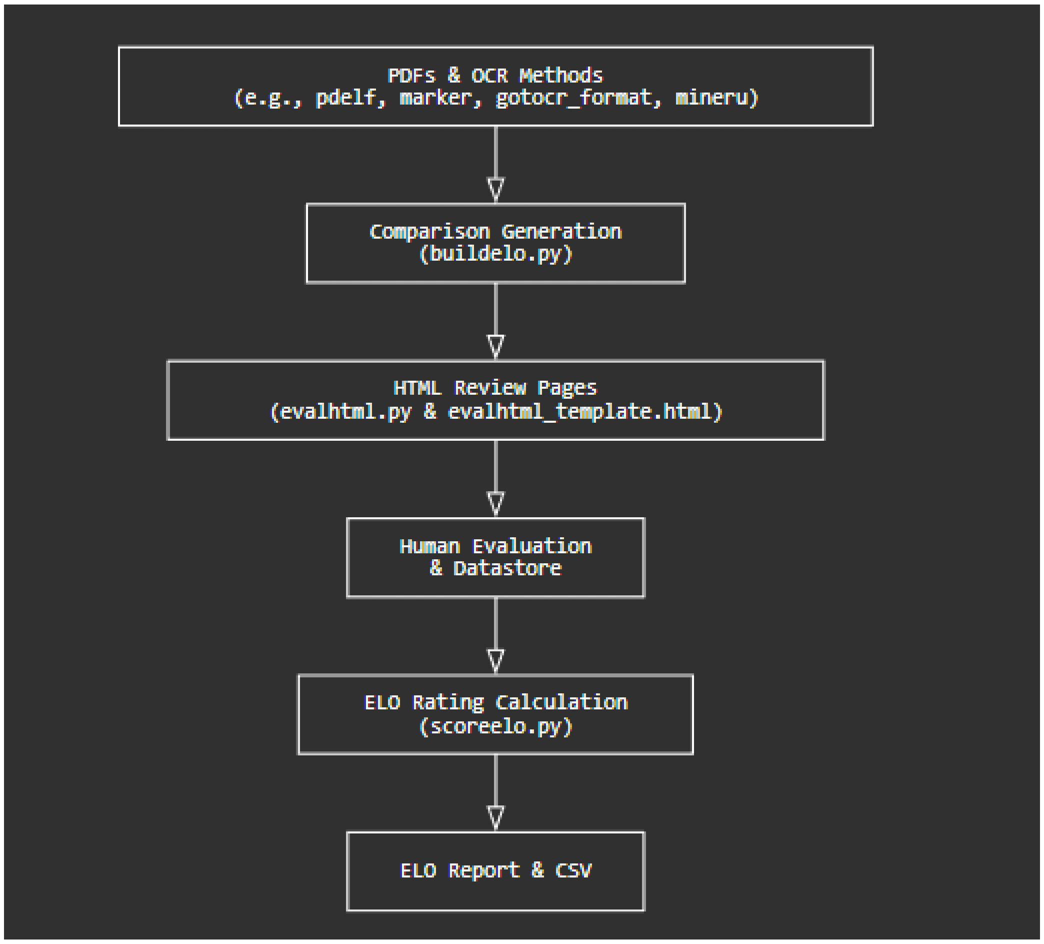
The methodology involves several key steps to ensure robust and statistically sound comparisons. First, win/loss records for different OCR methods are processed. These records indicate which OCR tool produced a "better" output for a given document. From this data, Elo ratings are calculated, providing a quantitative measure of each tool's performance.

To enhance the reliability of these ratings, a bootstrapping technique is applied. This involves resampling the input data multiple times to generate a distribution of Elo ratings for each method. From these distributions, 95% confidence intervals are derived, offering a range within which the true Elo rating is likely to fall. This statistical approach helps to account for variability in the judgment data.

Furthermore, pairwise significance tests are conducted between the Elo ratings of different OCR models. These tests determine whether the observed differences in performance between any two tools are statistically significant, rather than merely due to random chance. A statistically significant difference is often indicated by a p-value below a certain threshold (e.g., 0.05). The results, including bootstrapped Elo ratings with confidence intervals and pairwise significance test outcomes, are typically saved in a file such as **scripts/elo/results.txt**.

Finally, the results are visualized using boxplots, which provide a clear graphical representation of the Elo rating distributions for each OCR tool. These plots typically highlight **pdelf** (**olmOCR**) in a distinct color to facilitate direct comparison with other tools. This comprehensive approach, orchestrated by scripts like **scripts/elo/calculate_elo_ratings.py** and **scripts/elo/draw_boxplots.py**, allows for an objective and statistically rigorous assessment of OCR tool performance. For more details on the input data format used for these calculations, see Human-in-the-Loop Evaluation and Review Page Generation.

Human-in-the-Loop Evaluation and Review Page Generation



The olmocr project provides capabilities for human-in-the-loop evaluation of OCR methods, focusing on the generation of interactive HTML review pages, capturing user judgments, and subsequently analyzing this feedback to produce ELO ratings for different OCR tools. This process is orchestrated by scripts that prepare comparison data, render it into web pages, and then process the results of human evaluations.

The creation of human review pages begins with the script at scripts/eval/buildelo.py. This script identifies PDF documents and their corresponding Markdown text extractions generated by various OCR methods. It then generates pairwise comparisons between these methods for each PDF. To focus human effort on meaningful differences, comparisons with high text alignment scores, indicating minor distinctions, are filtered out. The alignment between text extraction outputs is calculated using components from scripts/eval/dolma_refine, which leverages algorithms like Hirschberg for sequence alignment and spaCy for text segmentation to determine the document edit similarity. The script then generates HTML review pages, presenting a curated set of these comparisons for human evaluation.

The script at **scripts/eval/evalhtml.py** is responsible for producing the HTML reports. It dynamically generates HTML diffs between the texts being compared, visually highlighting insertions and deletions. The associated PDF content for each comparison is also rendered into base64 encoded PNG images and embedded directly into the HTML page. This allows human evaluators to visually inspect the original document while assessing the quality of the OCR output.

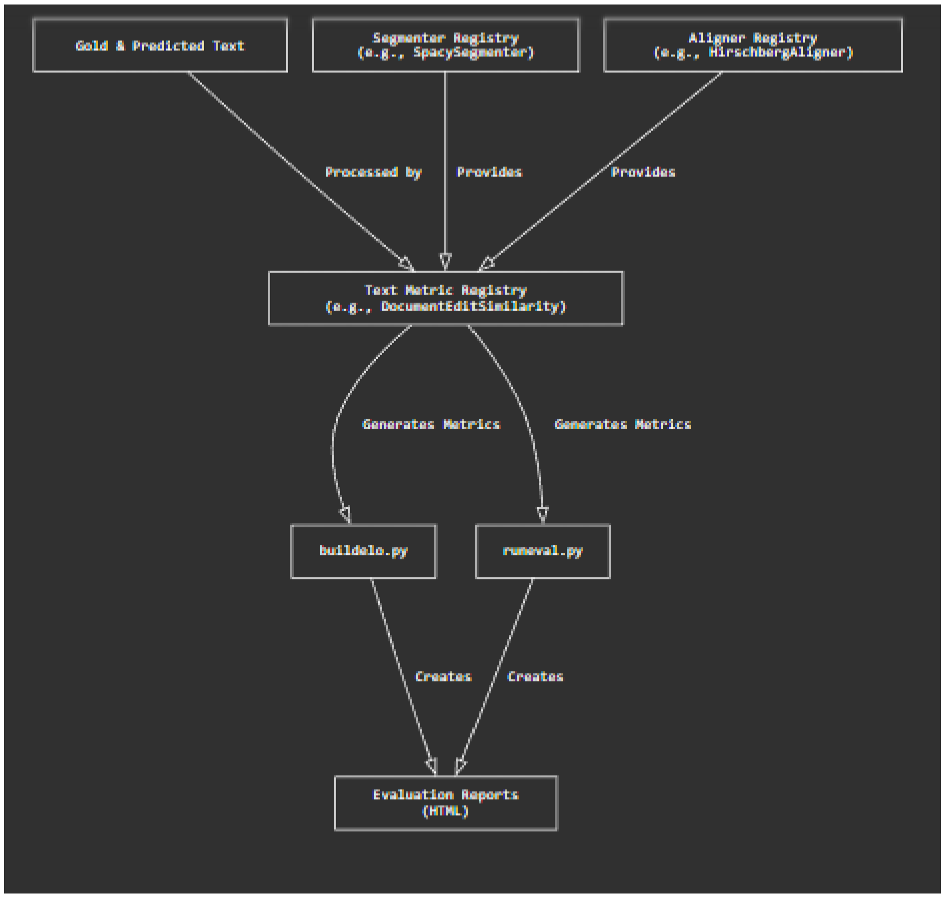
The interactive review interface is defined by the HTML template in **scripts/eval/evalhtml_template.html**. This template displays pairs of text blocks alongside the rendered PDF image. Users can select which text block they consider "better" or choose other conditions such as **both_good**, **both_bad**, or **invalid_pdf**. The interface incorporates client-side interactivity to manage user choices and maintains persistent state, allowing evaluators to return to their progress. It also includes controls for toggling diff views and revealing vote statistics.

After human evaluators have submitted their judgments, the script at **scripts/eval/scoreelo.py** analyzes this user review data. It retrieves the collected votes from the generated review pages and aggregates the head-to-head comparisons between different OCR methods. Based on these pairwise win/loss counts, the script calculates ELO ratings for each method. This ELO rating system provides an objective measure of relative performance, allowing for a ranked comparison of OCR tools based on human preferences. The results are presented as both a pairwise win/loss report and a final ELO ranking, which can be output to the console and saved as a CSV file.

An alternative evaluation path is provided by the script at **scripts/eval/runeval.py**, which compares the accuracy of OCR outputs against a "gold standard" dataset. Similar to **scripts/eval/buildelo.py**, this script performs detailed text alignment between the gold and evaluation texts to quantify similarity. It then generates HTML review pages for manual inspection, specifically highlighting the worst-performing comparisons and a random sample, ensuring that both problematic cases and representative samples are reviewed. This helps in understanding specific failure modes and overall performance trends.

Together, these components establish a robust framework for human-in-the-loop evaluation, combining automated text analysis with subjective human judgment to provide a comprehensive assessment of OCR system performance.

Text Extraction Evaluation and Alignment Framework


Gold & Predicted Text
Segmenter Registry (e.g., SpacySegmenter)
Aligner Registry (e.g., HirschbergAligner)
Processed by
Provides
Provides
Text Metric Registry (e.g., DocumentEditSimilarity)
Generates Metrics
Generates Metrics
buildelo.py
runeval.py
Creates
Creates
Evaluation Reports (HTML)

The olmocr project includes a comprehensive framework for evaluating text extraction from PDFs, focusing on advanced text similarity metrics and modular text processing components. This framework enables objective comparison of different OCR systems by quantifying the accuracy of their text output against a gold standard.

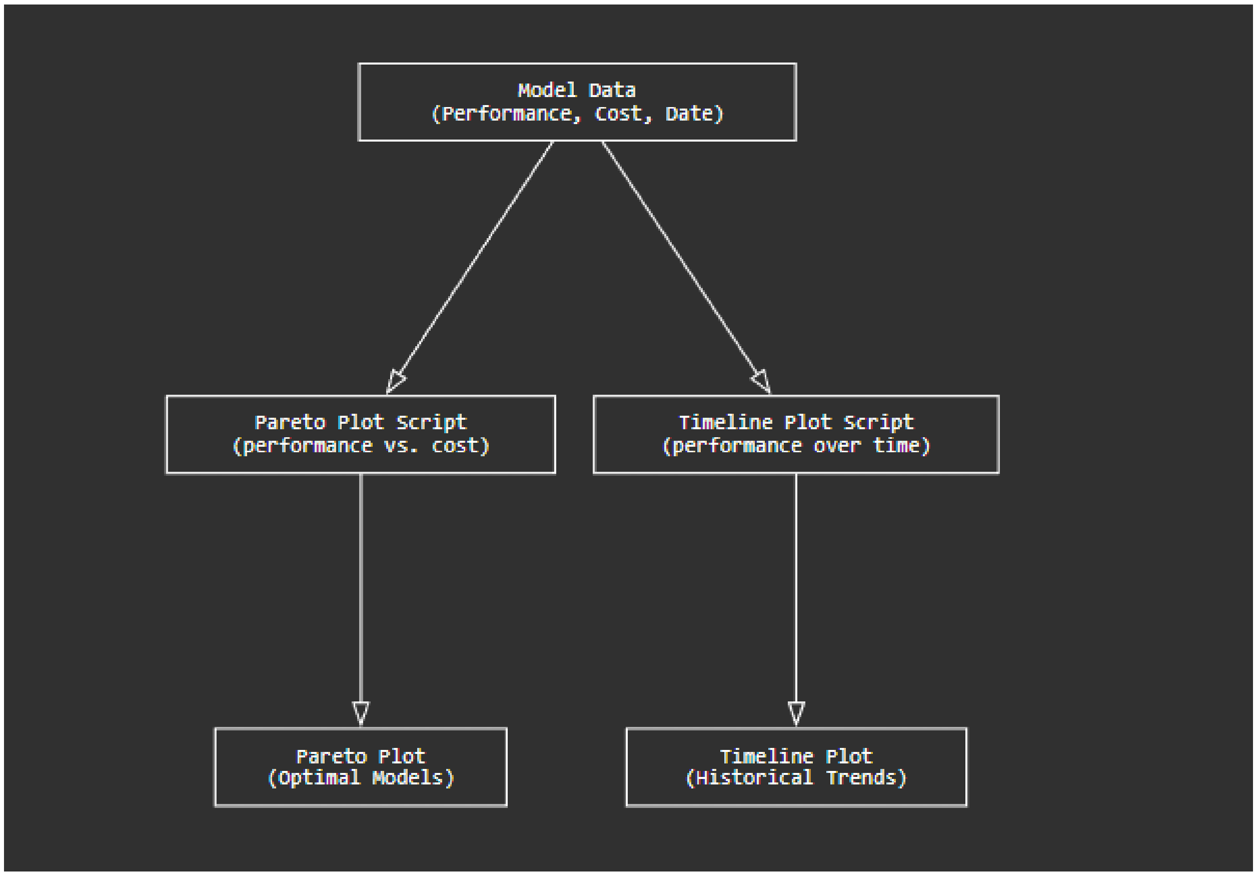
At the core of this evaluation system is the concept of edit similarity, applied at both the document and paragraph levels. DocumentEditSimilarity in scripts/eval/dolma_refine/metrics.py calculates a similarity score for entire documents by tokenizing and aligning the full text content. For more granular analysis, ParagraphEditSimilarity, also in scripts/eval/dolma_refine/metrics.py, extends this by focusing on specific segments of text, particularly those areas where significant differences ("gaps") exist between the gold and predicted texts. This approach allows for a targeted evaluation of challenging sections, rather than just an overall document score.

To achieve this precise text alignment and scoring, the framework utilizes modular components for segmentation and alignment. Text segmenters, managed by **SegmenterRegistry** in **scripts/eval/dolma_refine/segmenters.py**, break down continuous text into meaningful units, such as sentences. For instance, **SpacySegmenter** leverages the spaCy library for sentence-level segmentation. Once segmented, these text units are passed to aligners. The **AlignerRegistry** in **scripts/eval/dolma_refine/aligners.py** provides implementations of well-known sequence alignment algorithms like Hirschberg (**HirschbergAligner**) and Needleman-Wunsch (**NeedlemanWunschAligner**). These algorithms are crucial for determining the optimal sequence of edits (insertions, deletions, substitutions) required to transform one text sequence into another, thereby providing the foundation for calculating edit similarity scores.

The **TextMetricRegistry** in **scripts/eval/dolma_refine/metrics.py** serves as a central hub for managing these text similarity metrics, allowing for easy extensibility and dynamic selection of different evaluation approaches. This modular design ensures that new alignment algorithms, segmentation strategies, or similarity metrics can be integrated into the system without requiring significant changes to the existing codebase.

Beyond raw metric calculation, this framework is integrated into a broader evaluation pipeline. For example, the build process for Elo ratings, found in **scripts/eval/buildelo.py**, uses these similarity metrics to filter out near-identical comparisons before human review, focusing resources on more divergent cases. Similarly, the run evaluation script in **scripts/eval/runeval.py** leverages this framework to compare PDF conversion tactics against gold datasets, quantify performance, and generate review pages for further human inspection. This combination of automated metrics and human-in-the-loop validation provides a robust system for assessing the quality of text extraction. For more details on the human evaluation aspects, refer to Human-in-the-Loop Evaluation and Review Page Generation.

Visualizing OCR Model Performance Trade-offs and Evolution



The olmocr project provides capabilities for visualizing the performance of Optical Character Recognition (OCR) models through various plots. These visualizations are designed to illustrate performance trade-offs and the evolution of models over time, aiding in the comparative analysis of different OCR solutions.

One type of visualization is the Pareto frontier plot, which compares OCR model performance against their associated costs. This plot, generated by the script in scripts/plots/pareto_plot.py, helps identify models that offer optimal performance for a given cost or vice versa. The costs are calculated based on factors like GPU type, tokens processed per second, and pages processed per second, using predefined hourly costs for various GPUs. Models are categorized and visually differentiated on the plot to provide a clear comparison of various OCR engine types, including open-source tools and commercial Visual Language Models (VLMs).

Another key visualization is the timeline plot, which illustrates the performance evolution of OCR models over time. The script in **scripts/plots/plot_olmocr2_timeline.py** generates this plot, showcasing the progression of **olmOCR** versions, along with other OCR models, by plotting their overall performance scores against their release dates. This allows for an understanding of how model performance has improved or shifted historically. The plot uses distinct visual styles for different model categories to enhance readability and distinguish trends.

Both types of plots contribute to an objective understanding of OCR model capabilities, supporting decisions based on performance metrics and resource allocation.