

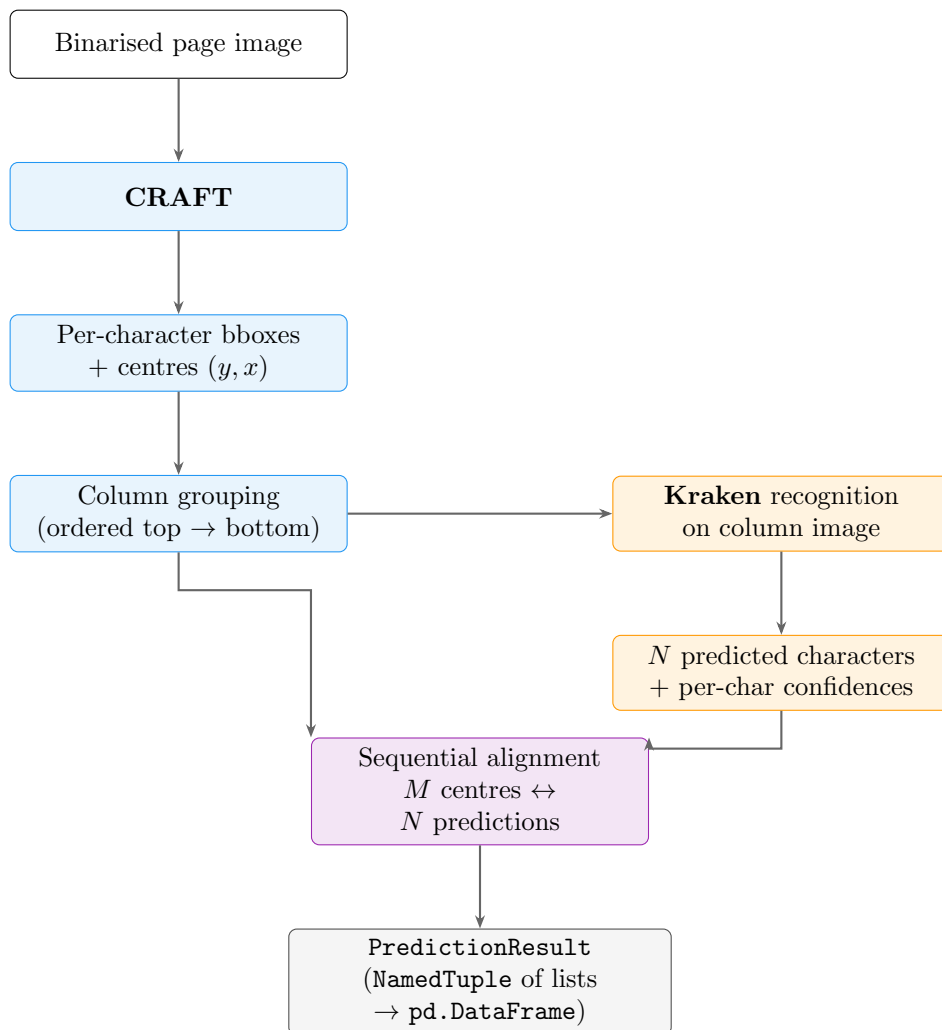
Character Recognition Pipeline

CRAFT → Kraken → Sequential Alignment

Overview

We decouple character **detection** (CRAFT) from character **recognition** (kraken). CRAFT provides reliable spatial localisation—per-character bounding boxes and centres—and is used to group characters into columns. Kraken receives each column image and predicts the character sequence. The two outputs are then aligned sequentially: one predicted character per CRAFT centre.

Pipeline



Kraken Recognition Model

Kraken is a line-level OCR engine. The recognition model (a VGSL-specified neural network, typically LSTM-based) expects a **single text-line image** as input. We pass the full column crop directly—no bounding-box extraction or baseline dewarping is performed by our code; the model's `ImageInputTransforms` handles all necessary resizing internally.

Internal processing:

1. `ImageInputTransforms`: resize to the network's expected height (preserving aspect ratio), with horizontal padding (`pad = 16` by default).
2. The network outputs a probability matrix over the character alphabet at each position along the column axis.
3. CTC decoding produces a list of (*char*, *start*, *end*, *confidence*) tuples.

We retain only the character identities and per-character confidences; spatial positioning comes from CRAFT.

Alignment

CRAFT detects M character centres; kraken predicts N characters. Both sequences follow the same reading order (top→bottom), so the alignment is **sequential**—character k from kraken maps to centre k from CRAFT.

When $N \neq M$:

Case	Handling	Logging
$N = M$	1:1 mapping	—
$N > M$	Truncate to first M predictions	Counter incremented; warning printed
$N < M$	Pad with (confidence = 0)	Counter incremented; warning printed

Running counters (`n_kraken_more`, `n_kraken_less`) track how often each case occurs across all columns. Call `wrapper.print_stats()` after processing to get the summary.

API Reference

Constructor

```
from model_wrapper import ModelWrapper

wrapper = ModelWrapper(
    rec_model_path="models/rec.mlmodel",
    device=None,      # auto-detect cuda/cpu
    pad=16,           # must match training config
)
```

`predict()`

```
result = wrapper.predict(
    image=column_img,                                     # (H, W) uint8 or bool
```

```

    bboxes=[(x0,y0,x1,y1), ...],          # M per-char bboxes
    centers=[(cy, cx), ...],              # M per-char centres (y,x)
)

```

Returns a `PredictionResult` (`NamedTuple`) with fields:

Field	Type	Description
<code>char</code>	<code>list[str]</code>	Predicted character (if unmatched)
<code>confidence</code>	<code>list[float]</code>	Per-character confidence (0.0 if unmatched)
<code>center_y</code>	<code>list[float]</code>	CRAFT centre y
<code>center_x</code>	<code>list[float]</code>	CRAFT centre x
<code>bbox_x0...bbox_y1</code>	<code>list[int]</code>	CRAFT bbox coordinates

Usage with Pandas

```

import pandas as pd

result = wrapper.predict(column_img, bboxes, centers)
df = pd.DataFrame(result._asdict())

#   char  confidence  center_y  center_x  bbox_x0 ...
#  0      0.998      45.0     120.0     110 ...
#  1      0.995      72.0     121.0     111 ...

```

Diagnostics

```

wrapper.print_stats()
# Total predict() calls: 38
#   Kraken > CRAFT:  2
#   Kraken < CRAFT:  1
#   Matched:         35

```