Extraction models

- When to use extraction models
- How extraction models work
- Extraction model capabilities
  - Extraction
  - Computation
  - Estimation
  - Forecasting
- Interpret extraction results
  - Missing results
  - Uncertainty
  - Constraints
  - Source spans
- Design extraction tasks
  - Define clear fields
  - Choose appropriate types and constraints
  - Include task-relevant context
  - Match safeguards to risk
- Limitations
- Next steps

Extraction models turn information into specific values your application can use, such as an invoice total, a due date, or an estimated payment amount. They return typed values rather than choosing from a fixed set of options or producing open-ended content. This helps you connect AI predictions to application logic.

You provide an input state and define the values you want the model to return, including each value's type and any additional constraints. The model uses these definitions and the input state to produce a corresponding typed result for each value. Typed results provide a predictable structure but do not guarantee correctness.

When to use extraction models

Use an extraction model when your application needs to extract, compute, estimate, or forecast a value, such as an amount, date, or duration. Alternatively, use a decision model to select from known options, or a generative model to produce open-ended content.

The following table compares these model types using invoice examples:

Model type	Use when you need	Invoice example
Extraction	Extract, compute, estimate, or forecast values, such as amounts, dates, durations, or text.	Read the total and due date from an invoice.
Decision	Select from a predefined set of options.	Determine if an invoice requires approval, manual review, or rejection.
Generative	Produce open-ended content.	Summarize discrepancies between the invoice, purchase order, and delivery receipt.

Your application can use returned values in application logic or pass them as inputs to another model.

How extraction models work

An extraction request combines an input state with a set of named field definitions. The input state supplies the information the model evaluates. Each field definition describes a value to return, specifies its type, and may include constraints. The model uses both to produce a corresponding result for each field.

For example, you provide invoice text that states a total of $1,080 and a due date of 2026-11-01. Separately, you define an invoice total field as a number in USD and a due date field as a date.

The model uses these definitions to identify the requested values in the invoice. It returns `1080` and `2026-11-01` as the corresponding typed results.

Your application can then use the returned values. For example, it can compare the due date with the current date to flag an overdue invoice.

Extraction model capabilities

Extraction models support extraction, computation, estimation, and forecasting.

Extraction

Return a value explicitly present in the input, such as the total printed on an invoice. Extraction identifies an existing value rather than calculating or estimating one.

Computation

Derive a value from facts in the input. For example, calculate how many days an invoice is overdue using its due date and a current date you provide. Unlike extraction, the result does not need to appear explicitly in the input.

Estimation

Predict a value that the input does not determine, such as when an unpaid invoice will be paid. Unlike computation, estimation addresses an unknown outcome rather than deriving a result from supplied facts.

Forecasting

Predict a sequence of future values over an interval, such as an invoice's outstanding balance each week for the next month. Unlike a single estimate, a forecast returns a value for each step in the requested period.

Interpret extraction results

Types and constraints define the form a result can take. They do not guarantee that it answers your question correctly.

Missing results

For fields that extract or compute a single value, a `null` result indicates that the model considers the input insufficient to determine the answer. The information may be missing, or the model may have failed to identify it.

Lists return an empty list when the model finds no matching values. Estimates and forecasts return predictions even when the input provides little information.

Uncertainty

The field type determines how uncertainty is represented and how it can be interpreted. Fields that extract and compute a single value include confidence scores for the returned answer. Estimates and forecasts include distributions over possible values.

For example, an estimated payment date might have a 10th percentile of November 5 and a 90th percentile of November 20. These dates mark the middle 80% of the predicted distribution. The actual payment date can still fall outside that interval.

Constraints

Constraints restrict the values the model can return. They do not guarantee that the answer is correct.

For example, restricting an estimated payment date to November ensures that the returned date falls in November. The invoice could still be paid in December.

Source spans

Extracted text includes a source span, which identifies its start and end positions in the input. You can use these positions to find the returned text in its original context.

Design extraction tasks

Define clear fields

Give each field a descriptive name and use its instructions to explain what value to return. For example, a field named `invoice_total` could have the instructions "Total amount due, including tax and shipping, after discounts." Specify its type and unit separately as a number in USD.

Choose appropriate types and constraints

Set each field's type according to how your application uses the result. For example, use a date for a payment deadline and a duration for days overdue.

Use the available field settings to specify precision and value bounds. For example, request two decimal places for an invoice amount. When bounding an estimated payment date, choose a window that reflects plausible payment dates, not just your preferred deadline.

Include task-relevant context

Include the information each task depends on in the input state. For example, calculating days overdue requires the invoice's due date and a current date you provide. State which date to use rather than assuming the model knows when the request runs.

Match safeguards to risk

Choose safeguards based on the consequences of an incorrect result. Displaying an estimated payment date may require fewer checks than using an extracted amount to authorize a payment.

Decide how your application handles missing or uncertain results. Before consequential or irreversible actions, check results against other evidence or require human review.

Limitations

- Typed output does not guarantee correctness. A result can satisfy its type and constraints while still answering the question incorrectly.
- Results depend on the available information and field definitions. Returning an estimate does not mean the input provides enough information to predict the outcome accurately.
- Uncertainty estimates are not guarantees. Confidence scores and prediction intervals do not replace application-level checks. An interval's actual coverage can vary across tasks and datasets.
- Capabilities vary by model. Consult the Clairic model documentation for supported inputs, field types, metadata, context window, and model-specific limitations.

Next steps

- Review the Clairic model reference for supported inputs, field types, model-specific limits, request and response schemas, and invoice examples.
- Consult the Workers AI REST API reference for the inference endpoint.
- Configure a Workers AI binding to access Workers AI from a Worker.
- Use the Workers AI REST API to authenticate and send requests over HTTP.

□□□ #2
□ □□□ □□□ 2026-11-01 05:27:05 UTC
□ □□□ □□□ 2026-11-01 05:27:07 UTC