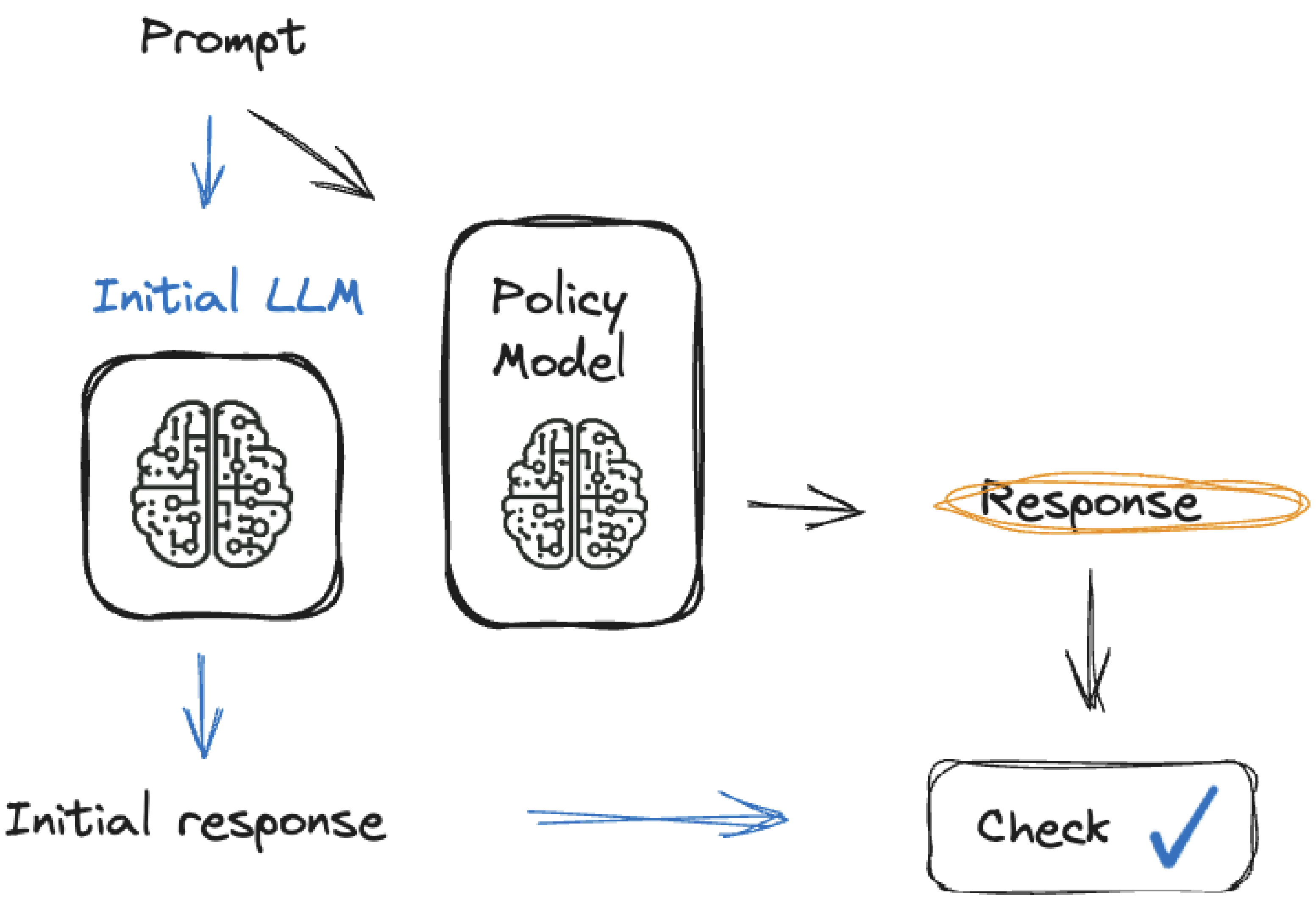
Methods for high-quality feedback gathering

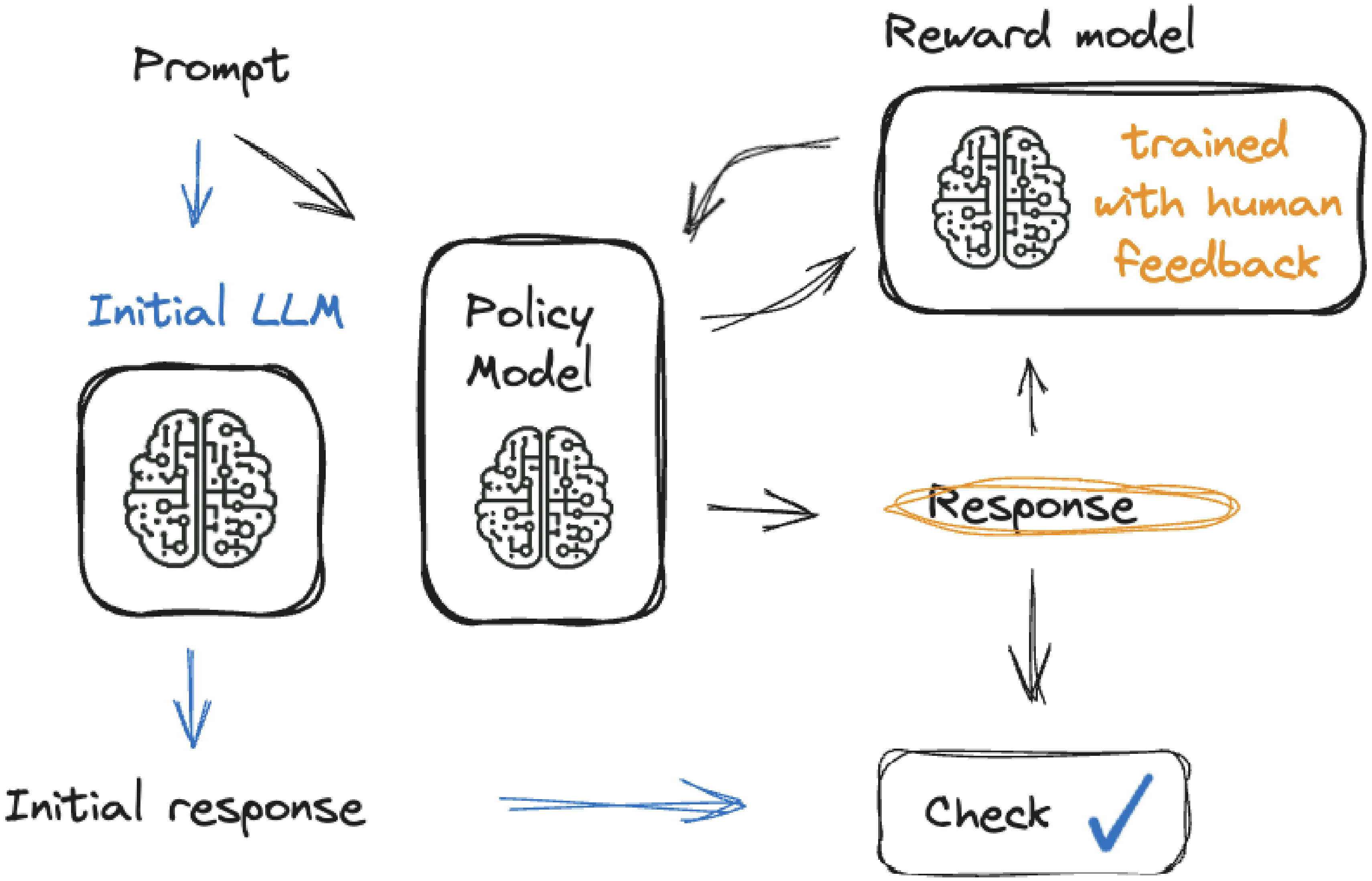
REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

Mina Parham
AI Engineer

Methods for high-quality feedback gathering



Methods for high-quality feedback gathering



Pairwise comparisons

- Choosing between two options:
- **Advantages:** Simple, intuitive, reduces bias
- **Challenges:** Provides less information per label
- **Example:** Movie A vs. Movie B: "Which do you prefer?

Pairwise comparisons

```python
def evaluate_responses(responses_A, responses_B):
    wins_A, wins_B = 0, 0
    for (response_A, score_A), (response_B, score_B) in zip(responses_A, responses_B):
        if score_A > score_B:
            wins_A += 1
        else:
            wins_B += 1
    success_rate_A = (wins_A / len(responses_A)) * 100
    success_rate_B = (wins_B / len(responses_B)) * 100
    return success_rate_A, success_rate_B
```

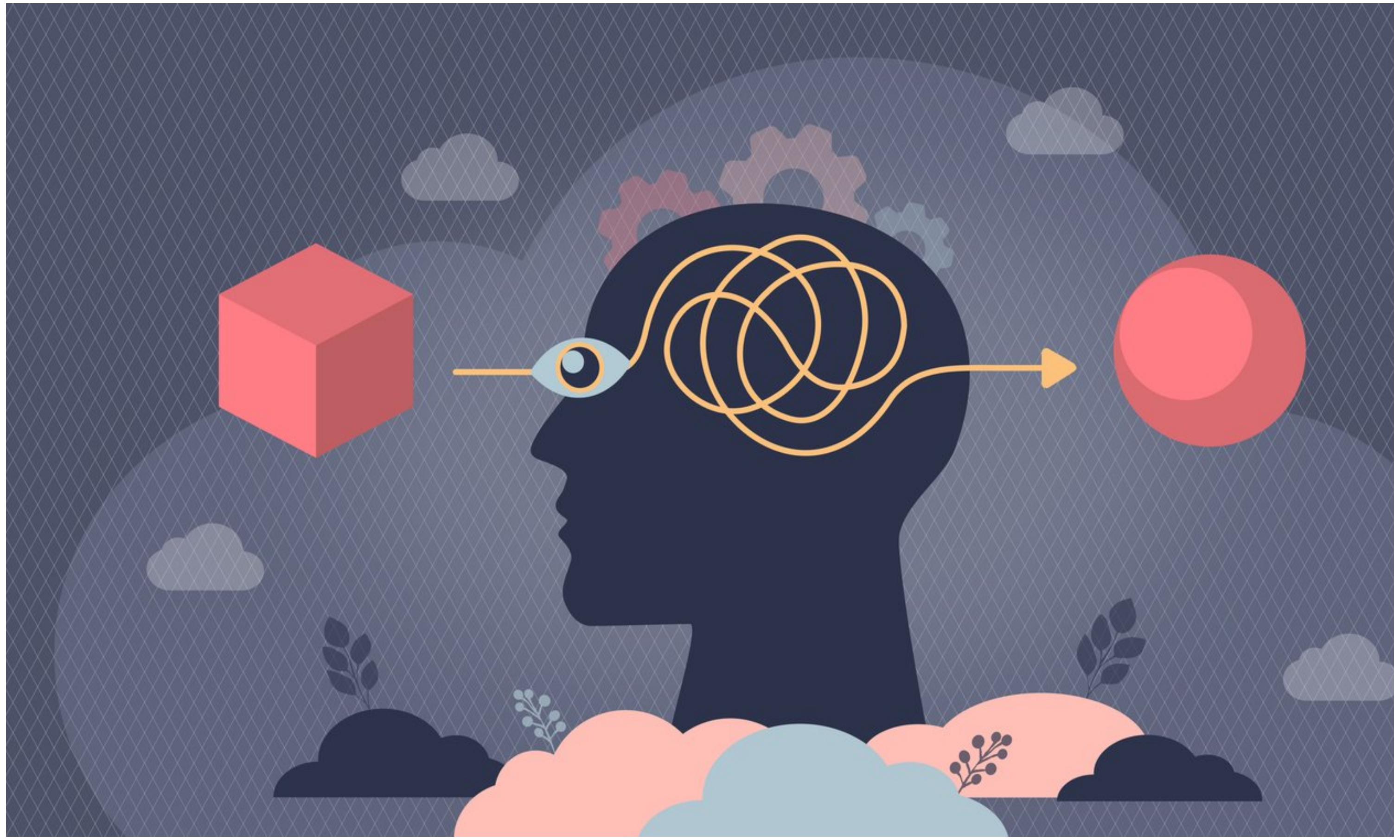
Ratings

- Assigning a score on a scale:
- **Advantages:** Provides more detailed feedback
- **Challenges:** Prone to biases, inconsistent scales
- **Example:**

```
Movie A: 4/5
Movie B: 3/5
```

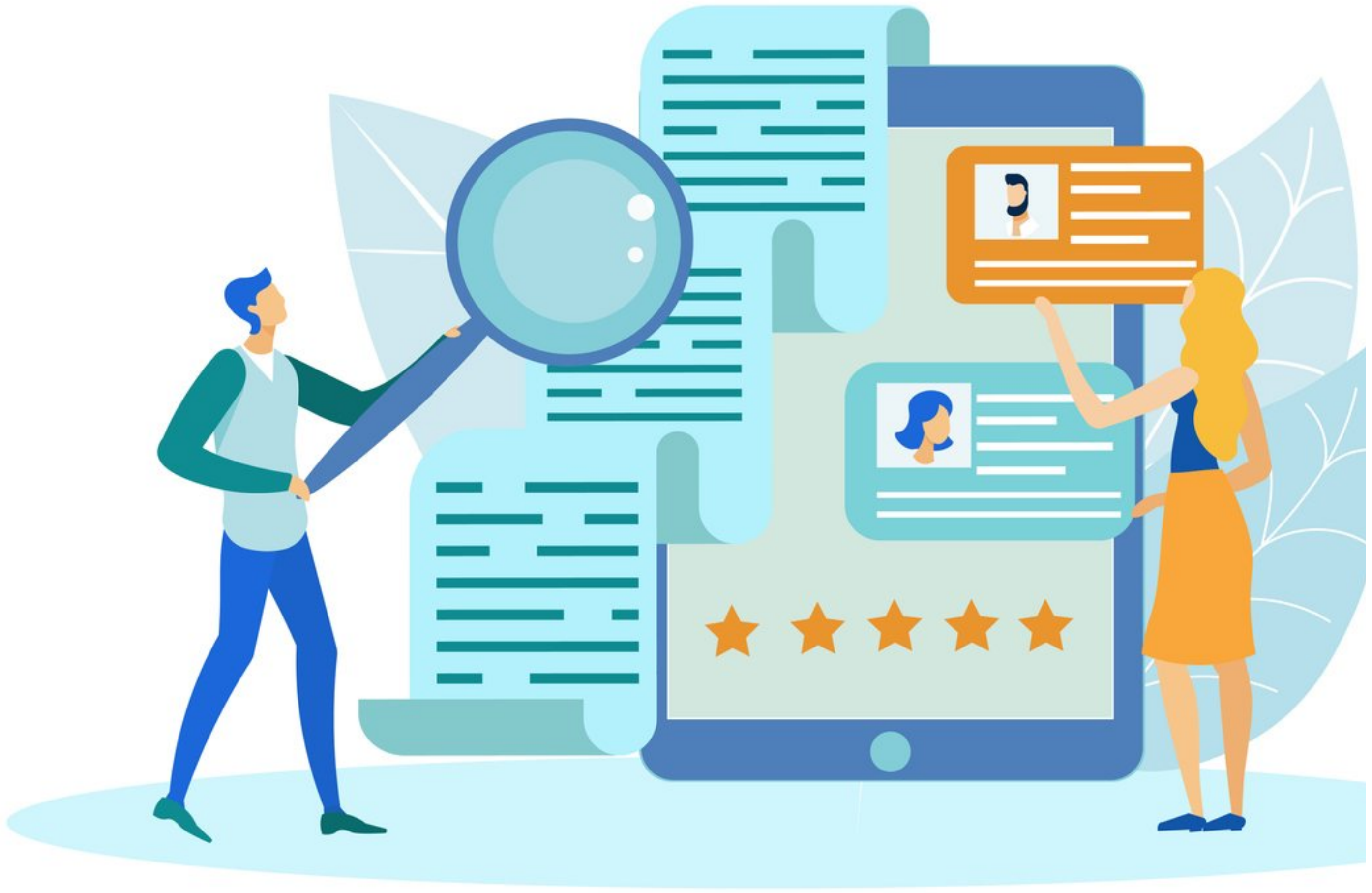
Psychological factors

- **Cognitive Biases:**
  - **Framing Effect:** How a question is presented can influence responses
  - **Serial Position Effect:** The order in which options are presented can affect decisions
  - **Anchoring:** Previous information biases current decisions



Guidelines for collecting high-quality feedback

- Cognitive load: tired users, inconsistent feedback
- **Carefully phrase questions**
  - To combat risks from cognitive load.
- **Randomize query order**
  - To minimize bias due to anchoring and framing
- **Collect Diverse Data**
  - To mitigate the issue of noise.



Let's practice!

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

Measuring feedback quality and relevance

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

Mina Parham
AI Engineer

Application of detecting anomalous feedback

For example:

- **Positive Review:**
  - "I loved this product!"
- **Negative Review:**
  - "Awful service."
- **Neutral Review:**
  - "Does what it's supposed to."
- **Outlier Review:**
  - "The sky is blue."

Detecting anomalous feedback

```
import numpy as np
```

```
def least_confidence(prob_dist):
    simple_least_conf = np.nanmax(prob_dist)
    num_labels = float(prob_dist.size)  # number of labels
    least_conf = (1 - simple_least_conf) * (num_labels / (num_labels - 1))
    return least_conf
```

```
def filter_low_confidence_predictions(prob_dists, threshold=0.5):
    filtered_indices = [i for i, prob_dist in enumerate(prob_dists)
                        if least_confidence(prob_dist) > threshold]
    return filtered_indices
```

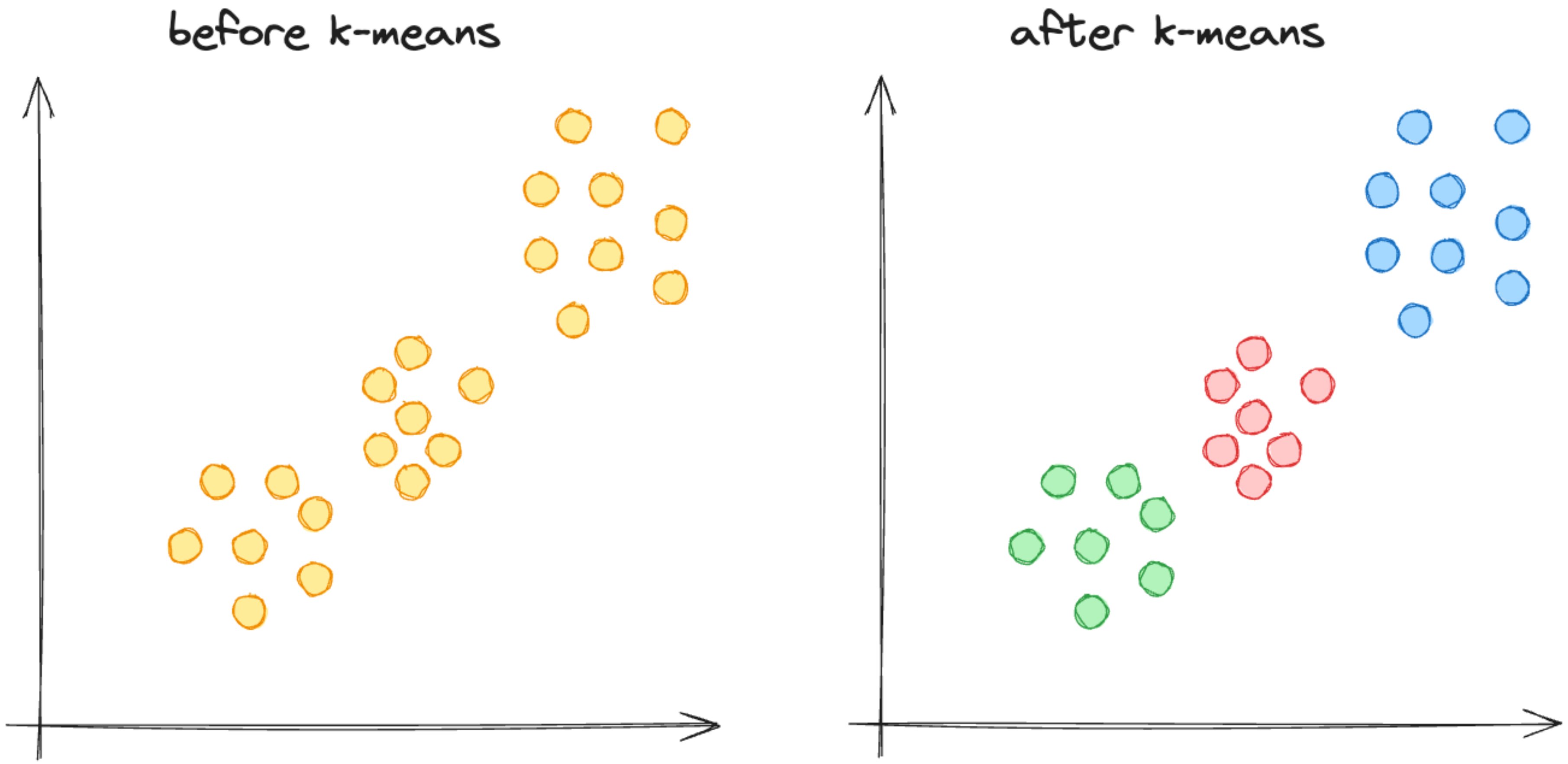
Detecting anomalous feedback

```
prob_distribution_array = np.array([
    [0.1, 0.1, 0.2],   # Low confidence (0.2)
    [0.6, 0.2, 0.1],   # High confidence (0.6)
    [0.3, 0.3, 0.4]   # Medium confidence (0.4)
])
# Filter function with 0.5 threshold
filtered_feedback_indices, filtered_confidences =
filter_low_confidence_predictions(prob_distribution_array, threshold=0.5)
print(f"Filtered Confidence Scores: {filtered_confidences}")
```

```
Filtered Confidence Scores: [0.6]
```

K-means

- Great for detecting anomalies and quick to implement
- Use domain knowledge or analytical methods to determine number of clusters



Anomaly detection with k-means

```python
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans

def detect_anomalies(data, n_clusters=3):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    clusters = kmeans.fit_predict(data)
    centers = kmeans.cluster_centers_
    # Calculate distances from cluster centers
    distances = np.linalg.norm(data - centers[clusters], axis=1)
    return distances
```

Anomaly detection with k-means

```
feedback_data = np.array([
    [4.0],  # Close to center of cluster
    [4.5],  # Close to center of cluster
    [1.0],  # Anomaly - far from main group
    [4.1],  # Close to center of cluster
    [3.9]  # Close to center of cluster
])
anomalies = detect_anomalies(confidences, n_clusters=1)
print(anomalies)
```

```
[0.5 1.  2.5  0.6 0.4]
```

Let's practice!

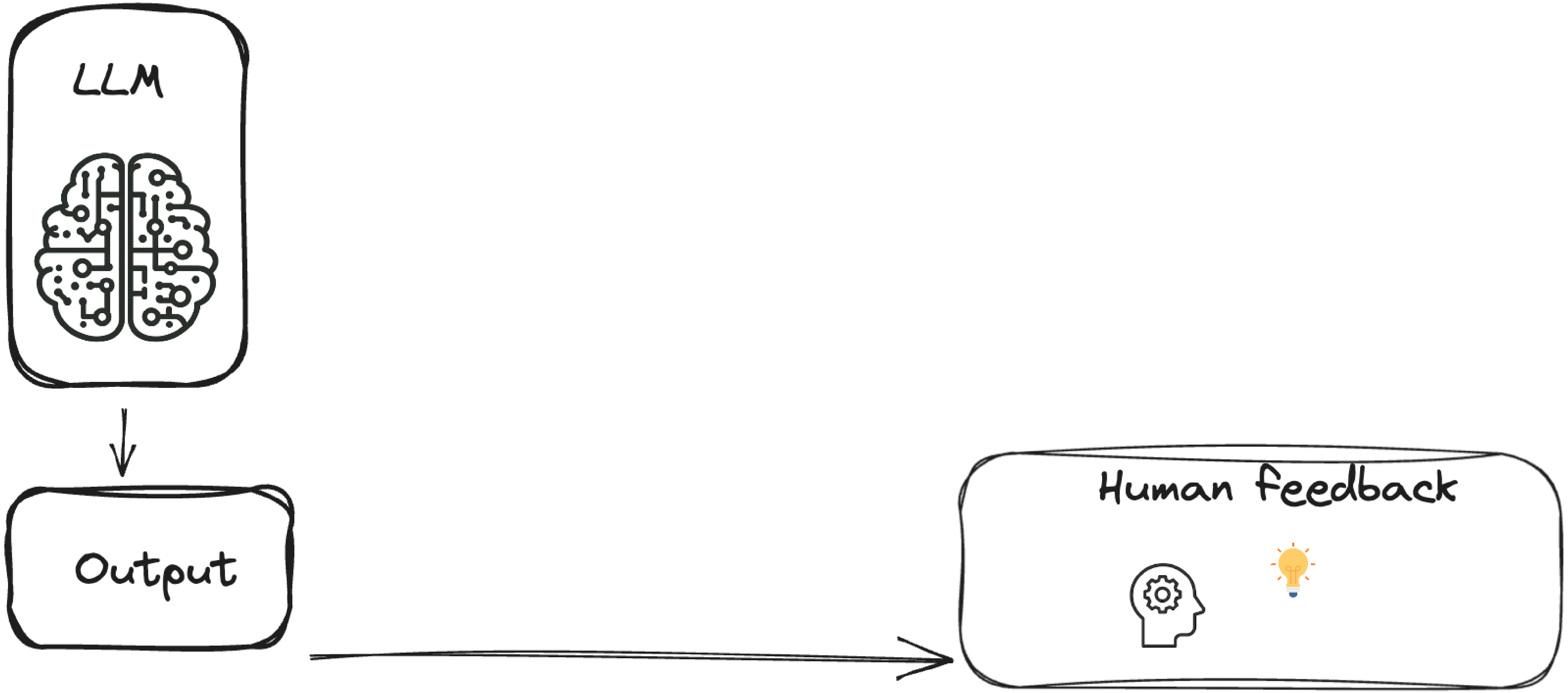
REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

Active learning

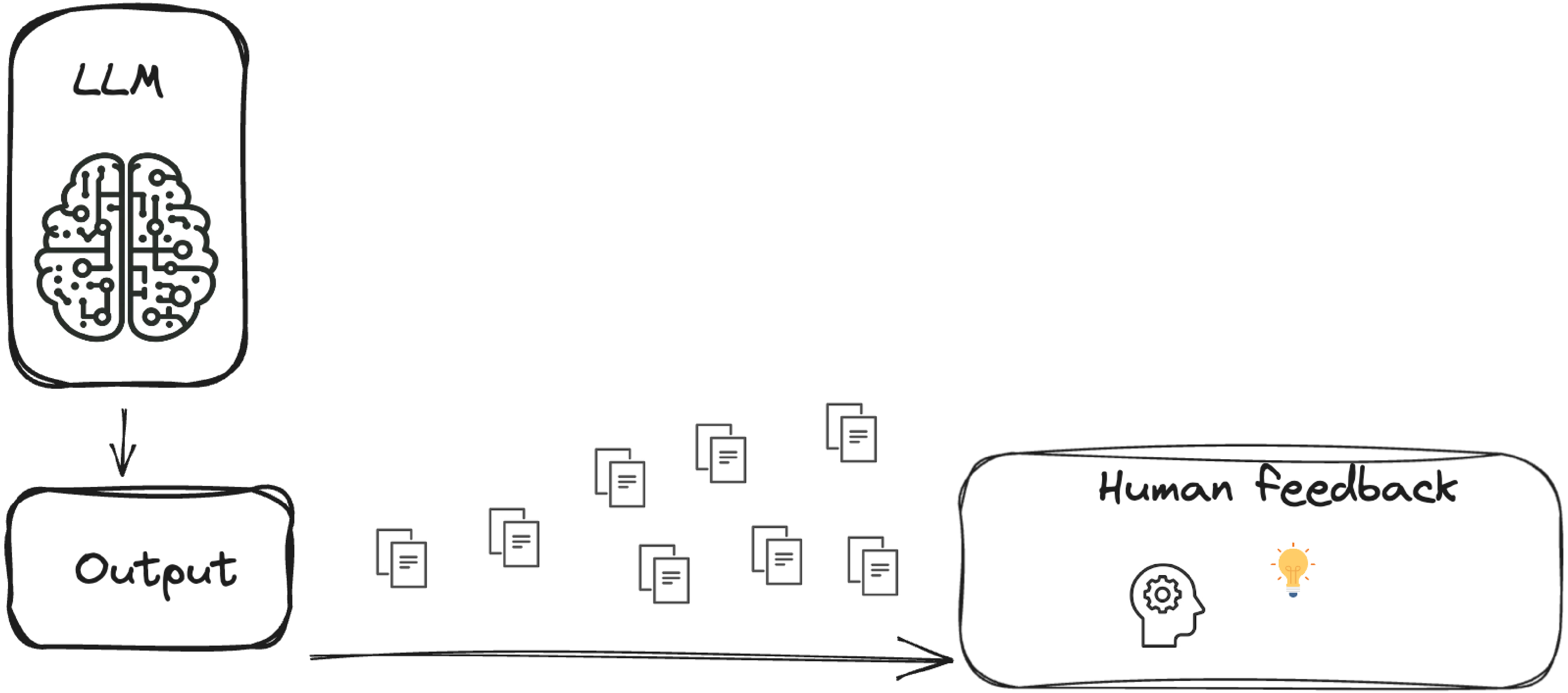
REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

Mina Parham
AI Engineer

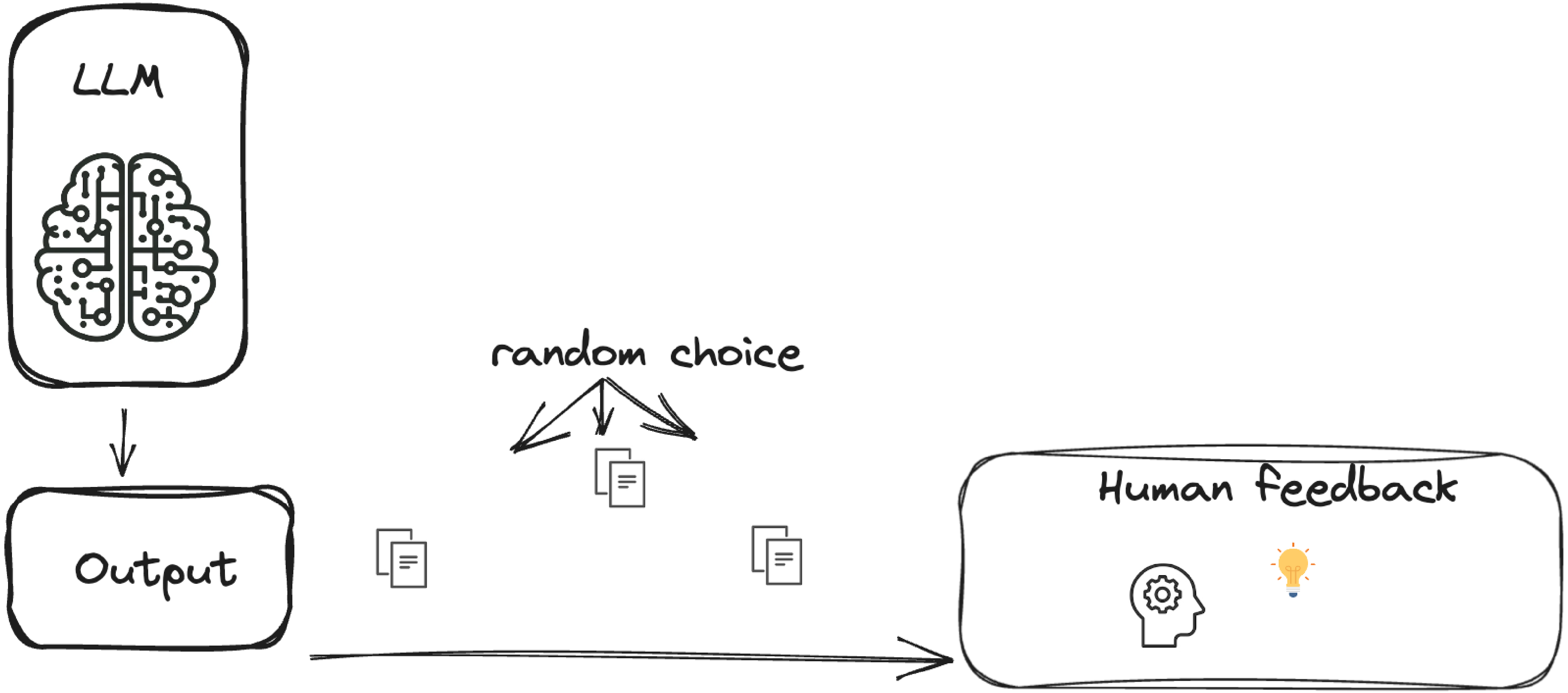
Human in the loop systems



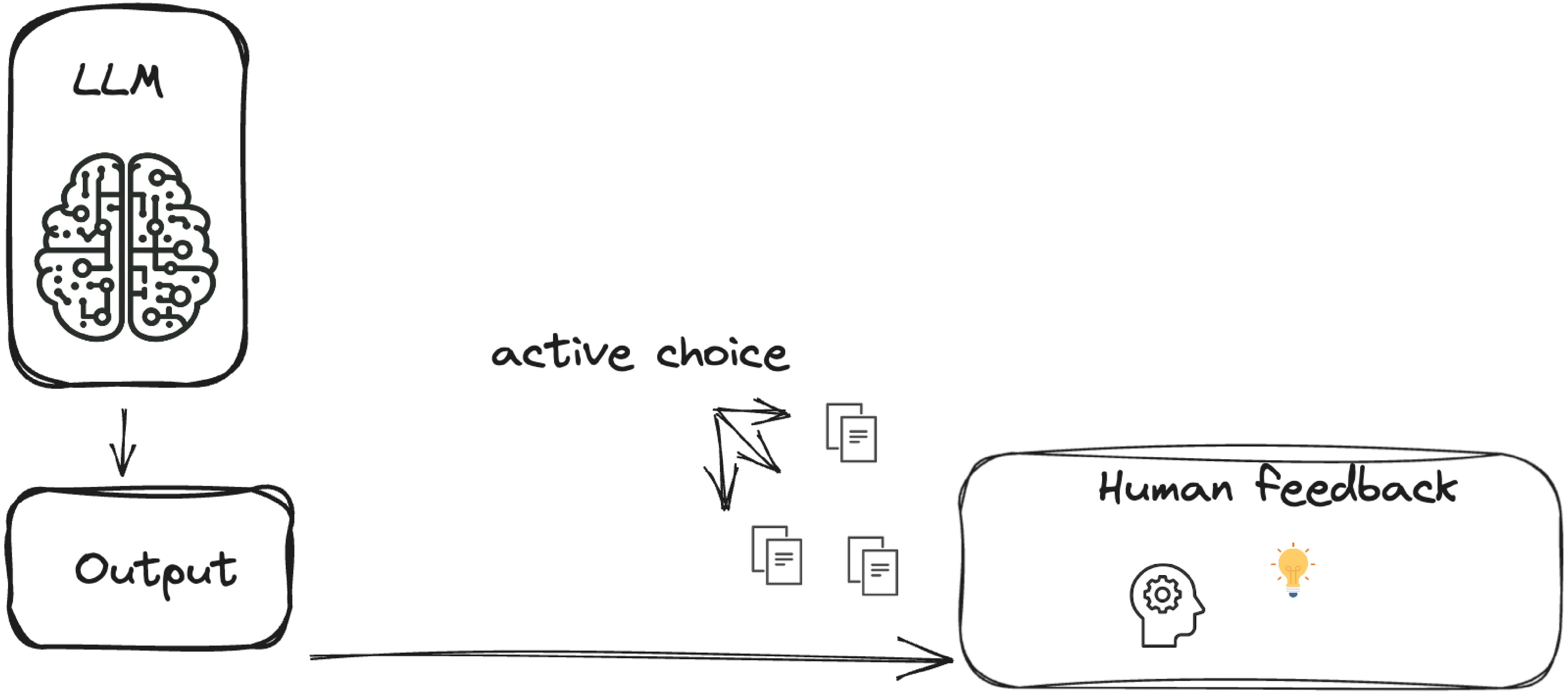
Human in the loop systems



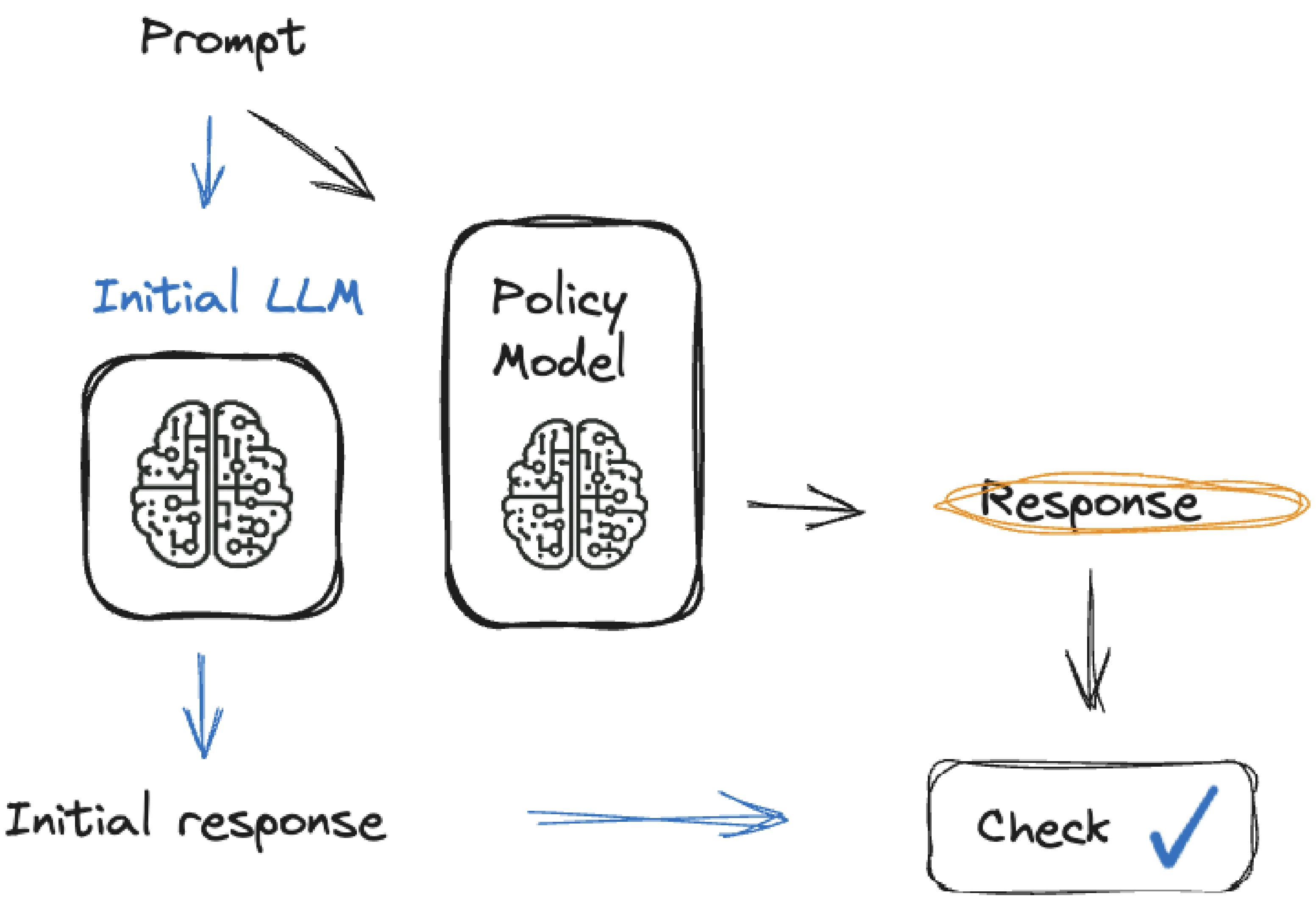
Human in the loop systems



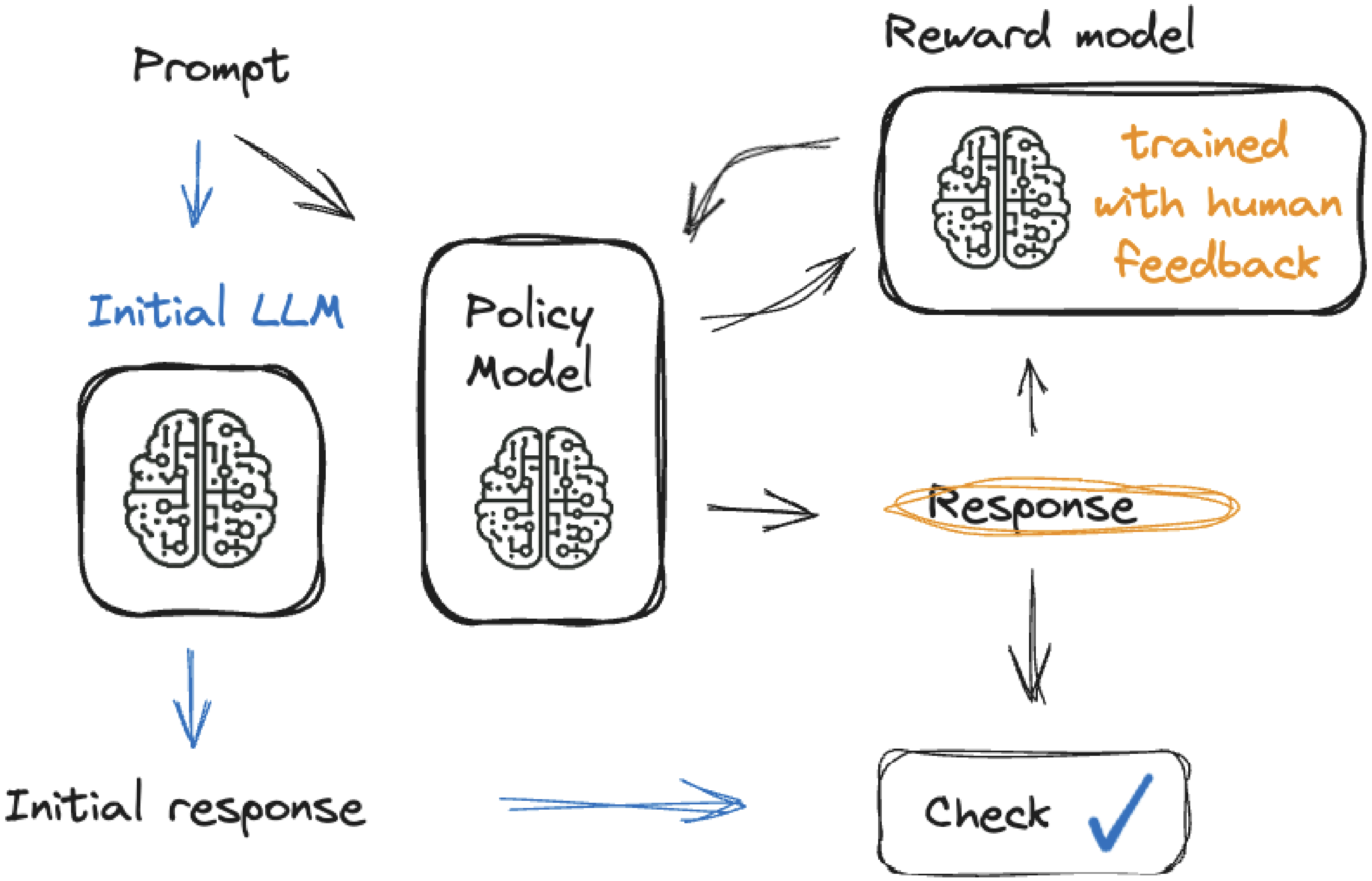
Human in the loop systems



Active learning in RLHF

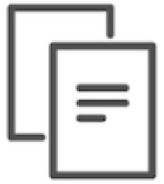


Active learning in RLHF

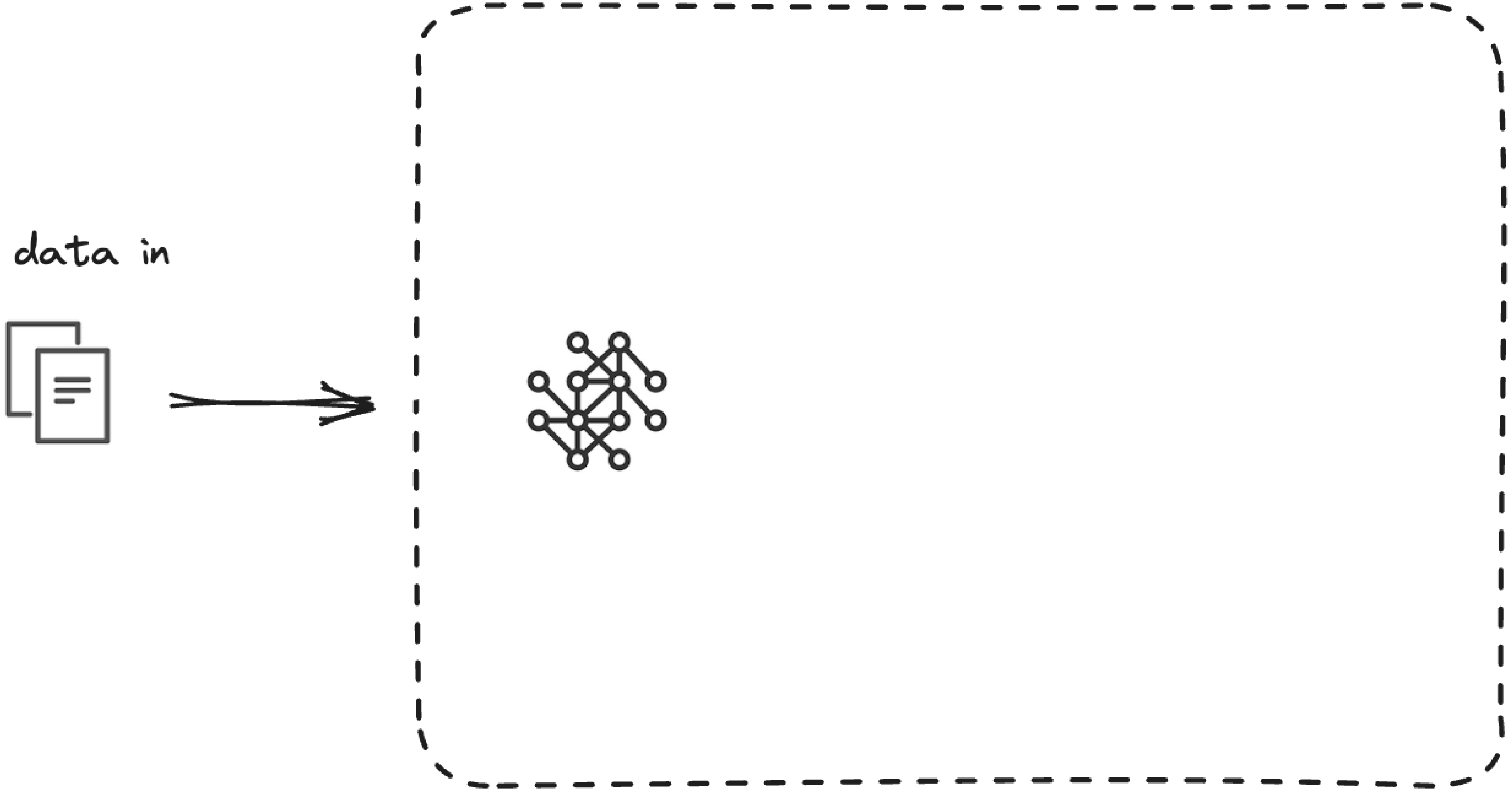


Active learning

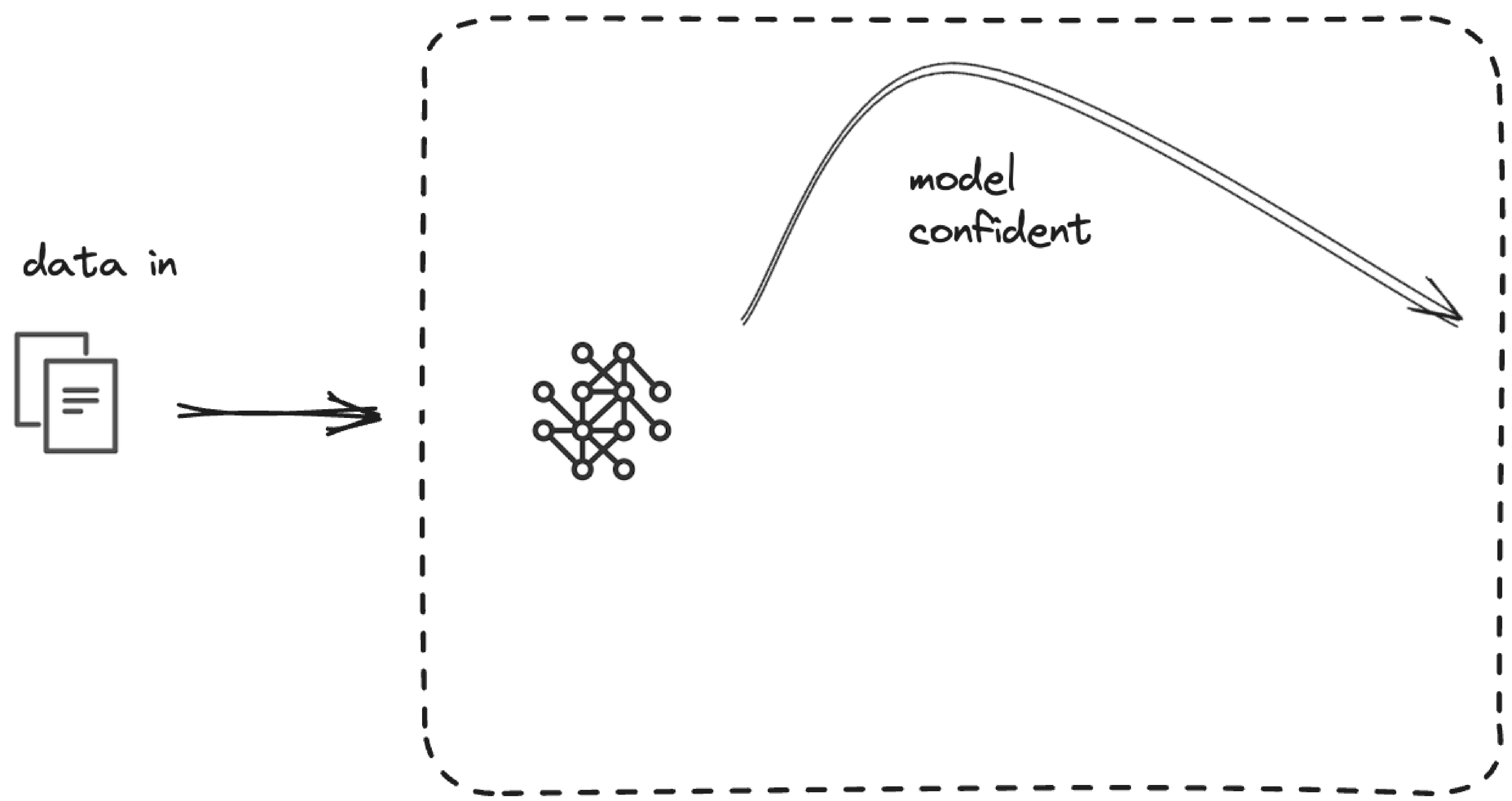
data in



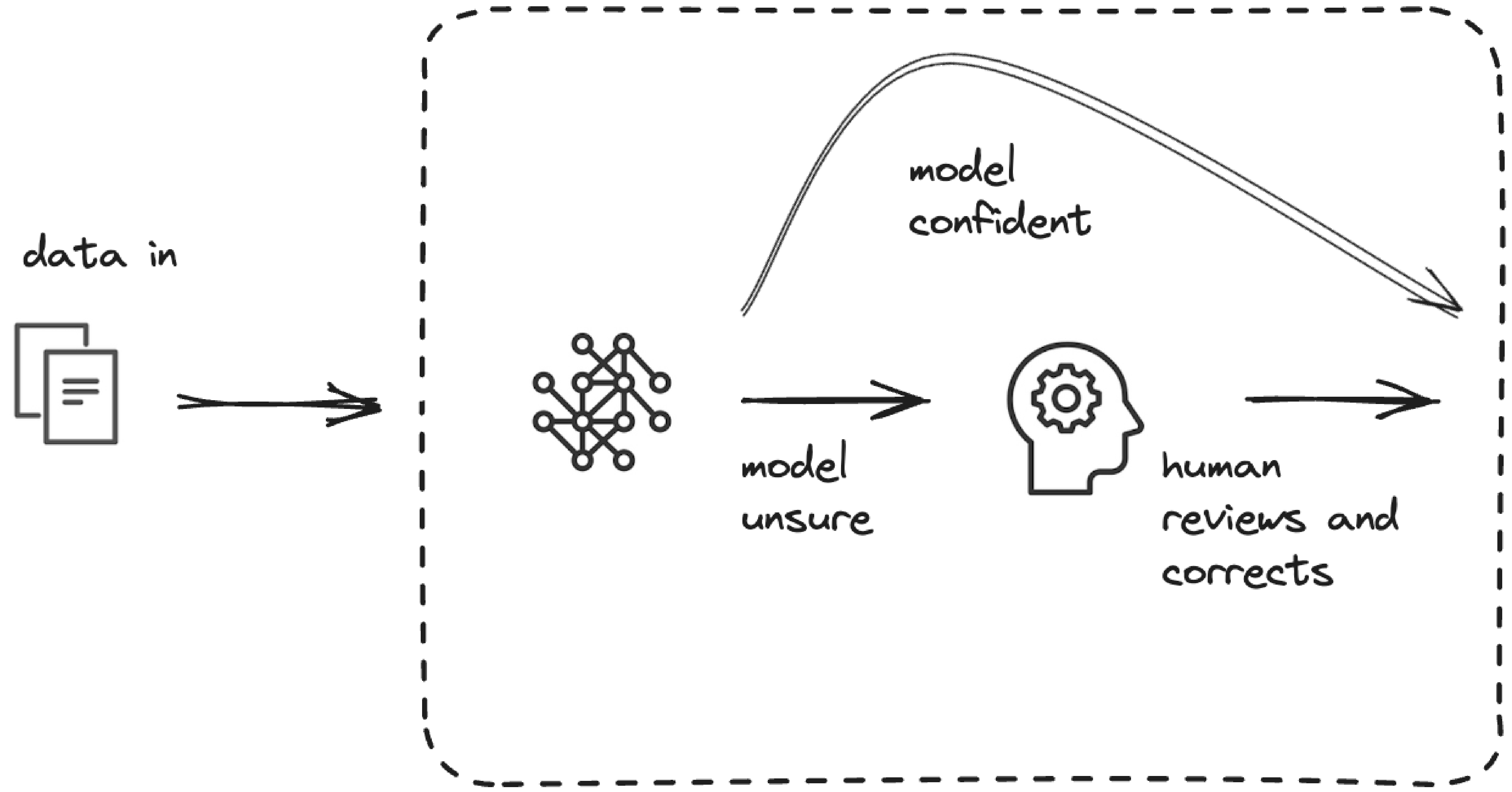
Active learning



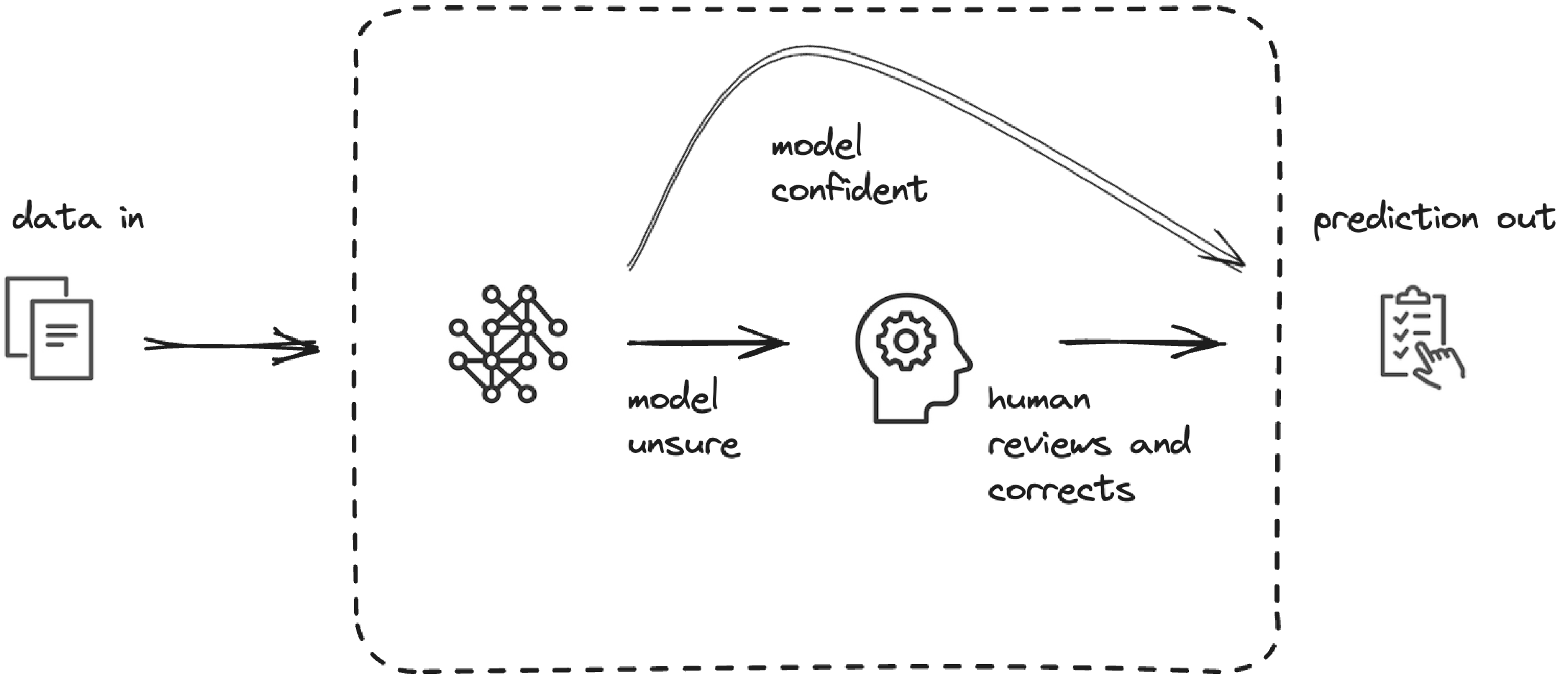
Active learning



Active learning



Active learning



Active learning pipeline with low confidence

```
from modAL.models import ActiveLearner
# Initialize learner
learner = ActiveLearner(
    estimator=LogisticRegression(),
    query_strategy=uncertainty_sampling,
    X_training=X_labeled, y_training=y_labeled
)
```

- **Uncertainty sampling:** points selected where confidence is lowest

Active learning pipeline with low confidence

```
# Active learning loop
for _ in range(10):
    learner.teach(X_labeled, y_labeled)
    query_idx, _ = learner.query(X_unlabeled)
    X_labeled = np.vstack((X_labeled, X_unlabeled[query_idx]))
    y_labeled = np.append(y_labeled, y[query_idx])
    X_unlabeled = np.delete(X_unlabeled, query_idx, axis=0)
```

Let's practice!

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)