```
import json

with open('dataset.json', 'r') as f:
    dataset = json.load(f)

print(dataset)
```

```
[{'user_input': "What's the weather like today?", 'intent': 'get_weather', 'entities': {'date': 'today'}}, {'user_input': 'Book a flight to New York tomorrow.',
```

```
[8] from transformers import pipeline

    intent_model = pipeline("zero-shot-classification")

    entity_model = pipeline("ner", aggregation_strategy="simple")
```

```
No model was supplied, defaulted to facebook/bart-large-mnli and revision c626438 (https://huggingface.co/facebook/bart-large-mnli).
Using a pipeline without specifying a model name and revision in production is not recommended.
No model was supplied, defaulted to dbmdz/bert-large-cased-finetuned-conll03-english and revision f2482bf (https://huggingface.co/dbmdz/bert-large-cased-finetune
Using a pipeline without specifying a model name and revision in production is not recommended.
Some weights of the model checkpoint at dbmdz/bert-large-cased-finetuned-conll03-english were not used when initializing BertForTokenClassification: ['bert.poole
- This IS expected if you are initializing BertForTokenClassification from the checkpoint of a model trained on another task or with another architecture (e.g. i
- This IS NOT expected if you are initializing BertForTokenClassification from the checkpoint of a model that you expect to be exactly identical (initializing a
```

```
[9] def extract_intent(user_input):
        labels = [entry['intent'] for entry in dataset]
        result = intent_model(user_input, candidate_labels=labels)
        return result

    def extract_entities(user_input):
        entities = entity_model(user_input)
        return entities
```

```
[10] feedback_store = []

     def collect_feedback(user_input, rating):
         feedback_store.append({"input": user_input, "rating": rating})

     def adjust_responses():
         for feedback in feedback_store:
             if feedback["rating"] < 3:
                 print(f"Improve response for: {feedback['input']}")
```

```
[11] def chatbot_response(user_input):
         # Extract intent and entities
         intent_result = extract_intent(user_input)
         entities_result = extract_entities(user_input)

         # Get the highest scored intent
         best_intent = intent_result['labels'][0]
         best_score = intent_result['scores'][0]

         # Extract entities
         entities = [{ "word": entity['word'], "score": entity['score'] } for entity in entities_result]

         # Generate a response based on intent (this is a placeholder)
         response = f"Intent: {best_intent}\nEntities: {entities}"

         # Return response to the user
         return response

     # Example usage
     user_input = "Please Cancel my 8AM appointment In Japan"
     response = chatbot_response(user_input)
     print(response)

     # Collect feedback
     rating = int(input("Rate the response from 1 to 5: "))
     collect_feedback(user_input, rating)
     adjust_responses()
```

```
Intent: cancel_alarm
Entities: [{'word': 'Japan', 'score': 0.9990658}]
Rate the response from 1 to 5: 5
```