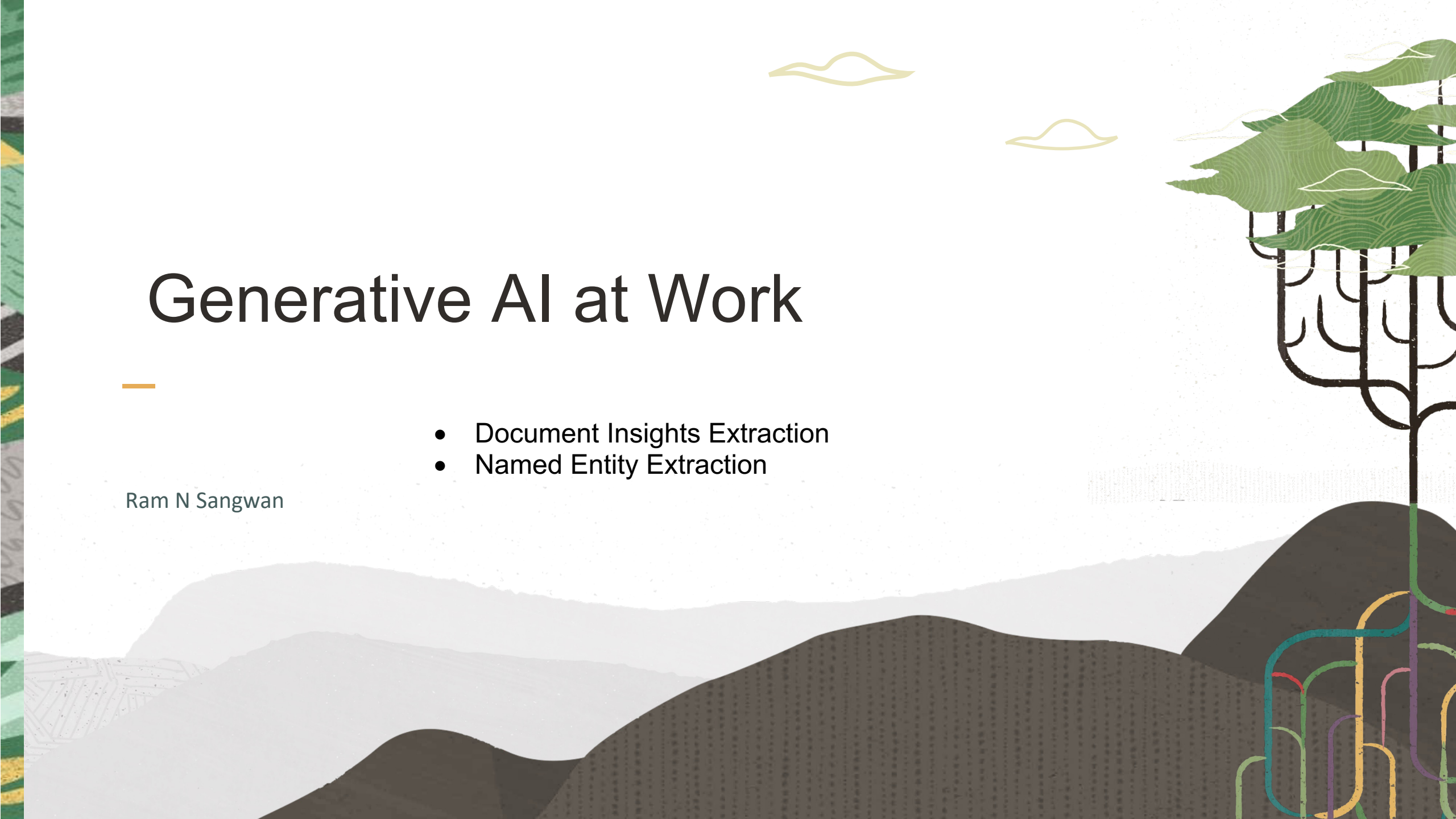
Generative AI at Work

- Document Insights Extraction
- Named Entity Extraction

Ram N Sangwan



Document Insights Extraction

- Summary & Entity Extraction using Generative AI

Example: Implementation Steps

Step 1: Choose the AI Model

- Select a pre-trained Generative AI model that has capabilities for natural language understanding and generation, such as ***Cohere Command*** or similar.

Step 2: Preprocess the Input Document

- **Text Extraction**: If the document is not in text format (e.g., PDF or Image), use an OCR (Optical Character Recognition) tool to convert it to text. Model may not be able to extract the text directly.
- **Data Cleaning**: Clean the text to remove any irrelevant sections or artifacts from OCR.

Step 3: Summarization

- **Summarization API Call**: Use the chosen AI model's summarization feature such as **Cohere Summarize** to generate a concise summary of the document.
- This might involve a specific API endpoint or a fine-tuned model for summarization.

Implementation Steps

Step 4: Entity Extraction

- **Entity Extraction API Call**: Use the same or a different AI model that's trained for Named Entity Recognition (NER) to extract entities from the text.
- **Customization**: Depending on the AI provider, you may need to fine-tune the model to improve entity recognition specific to the domain of your documents.

Step 5: Integration

- **API Integration**: Integrate both the summarization and entity extraction functionalities into a single workflow.
- **User Interface**: Develop a user-friendly interface where users can upload documents and receive insights.

Implementation Steps

Step 6: Postprocessing

- **Consolidation**: Merge the summary and the entities extracted into a coherent format, suitable for presentation.
- **Highlighting Entities**: Optionally, create a version of the summary where extracted entities are highlighted for easy reference.

Step 7: Output Presentation

- **Display Results**: Show the summary and entities to the user in the UI. This might be a web page or a software application.
- **Export Options**: Provide options to export the results, for example, as a PDF or a text file.

Implementation Steps

Step 8: Continuous Learning

- **Feedback Loop**: Implement a mechanism for users to provide feedback on the accuracy of the summary and entities extracted.
- **Model Retraining**: Use the feedback to retrain or fine-tune the model to improve accuracy over time.

Entity Extraction Tool Workflow

1. **User uploads a document** to the tool's interface.
2. The **document is converted to text** if needed using OCR.
3. **Text is preprocessed** to remove noise and prepare for analysis.
4. The **summary is generated** by the AI model, providing a condensed version of the document's content.
5. **Entities are extracted** by the AI model or a specialized NER tool, identifying names, organizations, dates, etc.
6. **Results are consolidated** and displayed, showing the summary and extracted entities.
7. **Users can provide feedback**, which is used for model improvement.

Example Models

- To perform document summary and entity extraction using Generative AI, you can utilize a combination of pre-trained models and libraries like BART (Bidirectional and AutoRegressive Transformers), HuggingFace's Transformers library, and spaCy for entity recognition.

Code Snippet

```
import torch
import spacy
from transformers import BartTokenizer, BartForConditionalGeneration


# Load pre-trained BART model and tokenizer
model_name = 'facebook/bart-large-cnn'
model = BartForConditionalGeneration.from_pretrained(model_name)
tokenizer = BartTokenizer.from_pretrained(model_name)


# Load spaCy model for entity recognition
spacy_model = 'en_core_web_sm'
nlp = spacy.load(spacy_model)
```

Code Snippet

```
# Function to generate document summary
def generate_summary(document):
  input_ids = tokenizer.encode(document, return_tensors='pt')
  output = model.generate(input_ids, num_beams=4, max_length=150, early_stopping=True)
  summary = tokenizer.decode(output[0], skip_special_tokens=True)
  return summary

# Function to extract entities
def extract_entities(document):
  doc = nlp(document)
  entities = [(ent.text, ent.label_)
```

Thank You