📚 Saluda: Cognitive Fingerprinting for Book Conversations

Building an AI Companion from Reading Memories

Collaborators: Amy Zhang, Claude AI, ChatGPT, Perplexity

Project Goal: Transform spontaneous book reactions into an AI companion that connects current emotions/thoughts to past reading experiences.

Status: 2025-10-04

This notebook demonstrates the **core proof-of-concept** for Saluda — an AI that remembers your book reactions and uses them for emotional support:

1. 📊 **Embedding & Chunking Pipeline**
   Convert manual book spelunking transcripts into searchable memory chunks.
2. 🗺️ **Cognitive Landscape Visualization**
   Map how different books cluster in your reading consciousness.
3. 💬 **Conversational AI Interface**
   Saluda responds to emotions by drawing on relevant reading memories.
4. 🔍 **Meta-Reasoning Analysis**
   Examine how Saluda thinks about making emotional connections.

🎯 Real-World Test: Addressing Today's AI Anxieties ⚠️

We deliberately tested Saluda on the **two biggest fears people have about AI right now**:

"The job market sucks right now" *(AI displacement anxiety)*

- Everyone's worried AI will take their jobs
- #1 concern in public AI discourse
- People feel powerless against technological change

"My parents won't listen to me" *(AI safety/youth concerns)*

- Parents struggling with kids navigating AI (deepfakes, misinformation, etc.)
- Generational divide on AI understanding
- **Young people lacking proper AI guidance/guardrails**

The Goal: AI could address the biggest fears about AI itself in a deeply personalized, thoughtful way. AI collaborators don't have to be part of the problem (disconnected, generic responses); perhaps they can help people process modern anxieties through their own authentic intellectual/emotional patterns.

```
# ====================================
# 0. Setup
# ====================================
# Install sentence-transformers if not already installed
!pip install -q sentence-transformers

# ====================================
# 1. Import Libraries
# ====================================
import pandas as pd
import numpy as np
from sentence_transformers import SentenceTransformer
import json
from typing import List, Dict

# ====================================
# 2. Load and Combine CSVs
# ====================================
csv_files = [
    'Risk.csv',
    'MinorFeelings.csv',
    'ReturnOfNative.csv'
]

# Read CSVs into DataFrames
dfs = [pd.read_csv(f) for f in csv_files]

# Fill missing values for author and book_title
for df in dfs:
    df['author'] = df['author'].fillna('Unknown')
    df['book_title'] = df['book_title'].fillna('Unknown')

# Combine all CSVs into one DataFrame
df = pd.concat(dfs, ignore_index=True)
print(f"Loaded {len(df)} rows from {len(csv_files)} files")
df.head()
```

```
Loaded 54 rows from 3 files
```

	book_title	author	edition	page	type	text	start_time	end_time
0	against the gods, the remarkable story of risk.	peter l. bernstein	NaN	1	passage	The revolutionary idea that defines the bound...	53.96	173.76
1	against the gods, the remarkable story of risk.	peter l. bernstein	NaN	1	reaction	Saluda, I like the clarity of the writing here...	NaN	NaN
2	against the gods, the	peter l.	NaN	1	passage	This book tells the story of a	193.60	233.94

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# ====================================
# 3. Define the Embedder Class
# ====================================
class BookSpelunkingEmbedder:
    def __init__(self, model_name: str = "all-MiniLM-L6-v2"):
        """Initialize with a lightweight sentence transformer model"""
        self.model = SentenceTransformer(model_name)
        self.chunks = []
        self.embeddings = None
        self.metadata = []

    def create_passage_reaction_chunks(self, df: pd.DataFrame) -> List[Dict]:
        """Create chunks by pairing passages with their reactions"""
        chunks = []

        for book_title in df['book_title'].unique():
            book_data = df[df['book_title'] == book_title].copy()
            author = book_data['author'].iloc[0]
            book_data = book_data.sort_values(['page', 'start_time'], na_position='l

            i = 0
            while i < len(book_data):
                current_row = book_data.iloc[i]

                if current_row['type'] == 'passage':
                    if (i + 1 < len(book_data) and
                        book_data.iloc[i + 1]['type'] == 'reaction' and
                        book_data.iloc[i + 1]['page'] == current_row['page']):

                        reaction_row = book_data.iloc[i + 1]
                        chunk_text = self._format_passage_reaction_chunk(
```

```
                            book_title, author, current_row['page'],
                            current_row['text'], reaction_row['text']
                        )

                        chunks.append({
                            'text': chunk_text,
                            'book_title': book_title,
                            'author': author,
                            'page': current_row['page'],
                            'chunk_type': 'passage_reaction_pair',
                            'passage_text': current_row['text'],
                            'reaction_text': reaction_row['text']
                        })
                        i += 2
                    else:
                        chunk_text = self._format_standalone_chunk(
                            book_title, author, current_row['page'],
                            'passage', current_row['text']
                        )
                        chunks.append({
                            'text': chunk_text,
                            'book_title': book_title,
                            'author': author,
                            'page': current_row['page'],
                            'chunk_type': 'standalone_passage',
                            'content': current_row['text']
                        })
                        i += 1

                elif current_row['type'] == 'reaction':
                    chunk_text = self._format_standalone_chunk(
                        book_title, author, current_row['page'],
                        'reaction', current_row['text']
                    )
                    chunks.append({
                        'text': chunk_text,
                        'book_title': book_title,
                        'author': author,
                        'page': current_row['page'],
                        'chunk_type': 'standalone_reaction',
                        'content': current_row['text']
                    })
                    i += 1
                else:
                    i += 1

        return chunks

    def _format_passage_reaction_chunk(self, book_title, author, page, passage, reac
        return f"""Book: {book_title} by {author}
Page: {page}
```

```
Passage: "{passage.strip()}"


My reaction: {reaction.strip()}"""

    def _format_standalone_chunk(self, book_title, author, page, content_type, conte
        type_label = "Passage" if content_type == "passage" else "My reaction"
        content_text = f'"{content.strip()}"' if content_type == "passage" else cont
        return f"""Book: {book_title} by {author}
Page: {page}


{type_label}: {content_text}"""

    def embed_chunks(self, chunks: List[Dict]) -> np.ndarray:
        texts = [chunk['text'] for chunk in chunks]
        embeddings = self.model.encode(texts, show_progress_bar=True)
        self.chunks = chunks
        self.embeddings = embeddings
        self.metadata = [{k: v for k, v in chunk.items() if k != 'text'} for chunk i
        return embeddings

    def save_embeddings(self, save_path: str):
        save_data = {
            'chunks': self.chunks,
            'embeddings': self.embeddings.tolist(),
            'metadata': self.metadata,
            'model_name': self.model.get_sentence_embedding_dimension()
        }

        # Convert all numpy.int64 or numpy.float64 to native Python types
        def convert_numpy(o):
            if isinstance(o, (np.integer, np.int64)):
                return int(o)
            elif isinstance(o, (np.floating, np.float64)):
                return float(o)
            else:
                return o

        with open(save_path, 'w') as f:
            json.dump(save_data, f, default=convert_numpy, indent=2)

        print(f"Saved {len(self.chunks)} chunks to {save_path}")

    def preview_chunks(self, n: int = 3):
        for i, chunk in enumerate(self.chunks[:n]):
            print(f"=== CHUNK {i+1} ===")
            print(chunk['text'])
```

```
            print(f"Type: {chunk['chunk_type']}\n")


# ====================================
# 4. Run the Workflow
# ====================================
# Initialize the embedder
embedder = BookSpelunkingEmbedder()

# Create chunks
chunks = embedder.create_passage_reaction_chunks(df)
embedder.chunks = chunks
print(f"Created {len(chunks)} chunks")

# Preview first 3 chunks
embedder.preview_chunks(3)

# Generate embeddings
embeddings = embedder.embed_chunks(chunks)
print(f"Generated embeddings with shape: {embeddings.shape}")

# Save to JSON
embedder.save_embeddings('book_spelunking_embeddings.json')
```

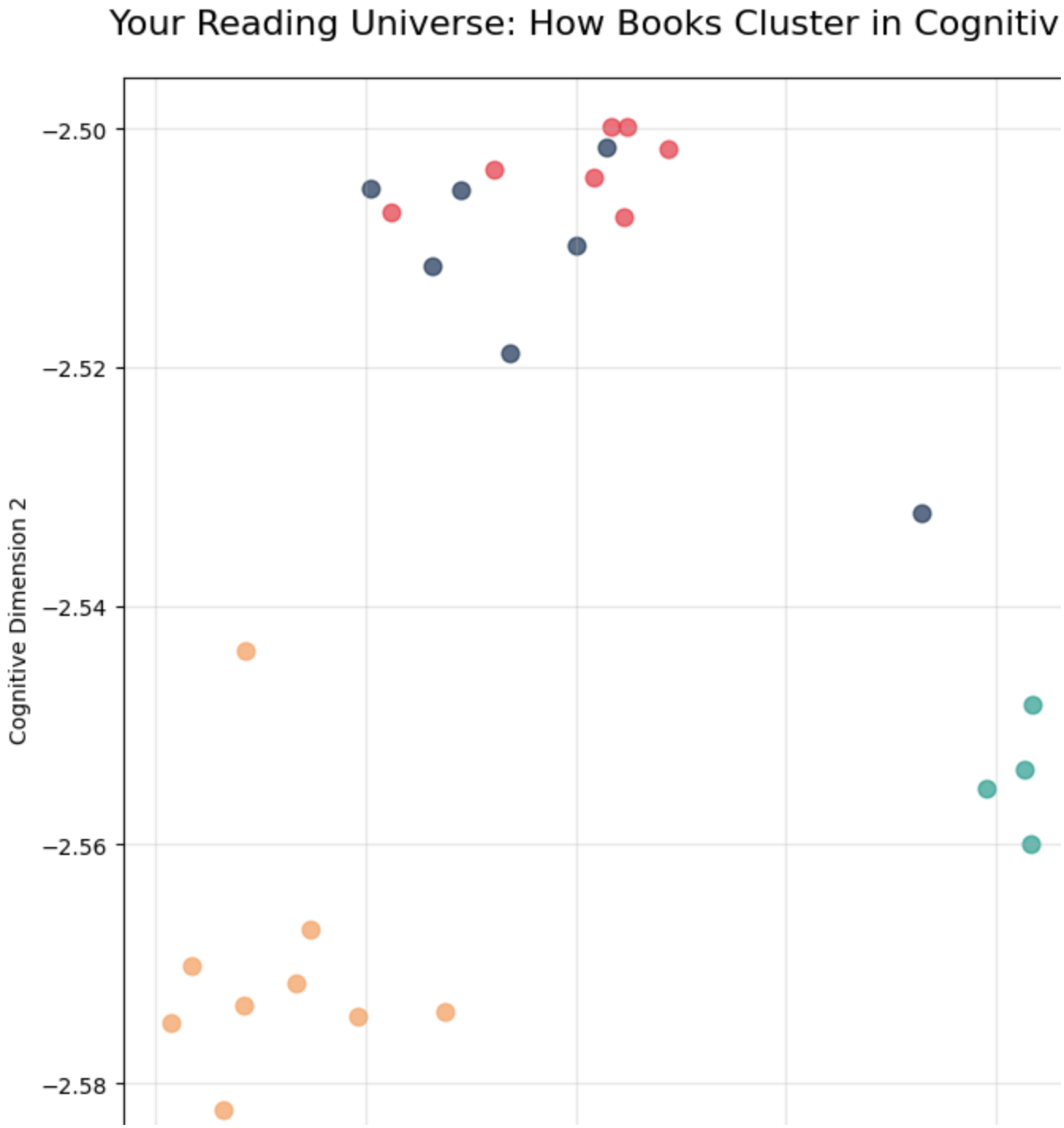
```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserW
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access publ
  warnings.warn(
Created 31 chunks
=== CHUNK 1 ===
Book: against the gods, the remarkable story of risk. by peter l. bernstein
Page: 1


Passage: "The revolutionary idea that defines the boundary between modern times
Type: standalone_passage

=== CHUNK 2 ===
Book: against the gods, the remarkable story of risk. by peter l. bernstein
Page: 1


Passage: "This book tells the story of a group of thinkers whose remarkable visi


My reaction: Saluda, I can almost hear the trumpet fanfare behind this guy's wor
Type: passage_reaction_pair

=== CHUNK 3 ===
Book: against the gods, the remarkable story of risk. by peter l. bernstein
Page: 1


My reaction: Saluda, I like the clarity of the writing here and the boldness of
Type: standalone_reaction

Batches: 100%                                                 1/1 [00:09<00:00,  9.18s/it]
Generated embeddings with shape: (31, 384)
Saved 31 chunks to book_spelunking_embeddings.json
```

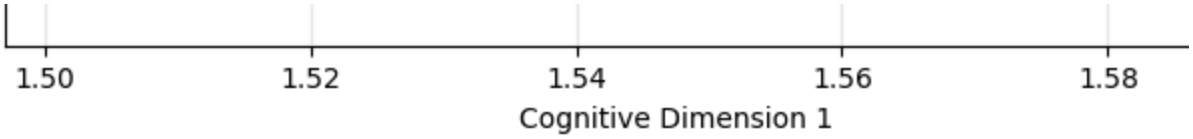
```
# Quick visualization of your reading embeddings in 2D space
import json
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
import seaborn as sns

# Load your embeddings
with open('book_spelunking_embeddings.json', 'r') as f:
    data = json.load(f)

embeddings = np.array(data['embeddings'])
chunks = data['chunks']
```

```
metadata = data['metadata']

print(f"Loaded {len(embeddings)} reading memories")

# Extract book titles for coloring
book_titles = [meta['book_title'] for meta in metadata]
unique_books = list(set(book_titles))
print(f"Books: {unique_books}")

colors = [
    '#E63946',  # red
    '#1D3557',  # dark blue
    '#2A9D8F',  # teal/green
    '#F4A261',  # orange
    '#264653',  # deep navy
    '#F94144',  # bright red
    '#577590',  # muted blue
]

book_colors = {book: colors[i % len(colors)] for i, book in enumerate(unique_books)}

# Reduce to 2D using t-SNE
print("Reducing embeddings to 2D... (this might take a moment)")
tsne = TSNE(n_components=2, random_state=42, perplexity=min(30, len(embeddings)-1))
embeddings_2d = tsne.fit_transform(embeddings)

# Create the plot
plt.figure(figsize=(12, 8))

# Plot each book with different colors
for book in unique_books:
    book_indices = [i for i, title in enumerate(book_titles) if title == book]
    book_embeddings = embeddings_2d[book_indices]

    plt.scatter(
        book_embeddings[:, 0],
        book_embeddings[:, 1],
        c=book_colors[book],
        label=book,
        alpha=0.7,
        s=60
    )

plt.title("Your Reading Universe: How Books Cluster in Cognitive Space", fontsize=16
plt.xlabel("Cognitive Dimension 1")
plt.ylabel("Cognitive Dimension 2")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, alpha=0.3)
plt.tight_layout()

# Show some insights
```

```
print("\n=== Quick Insights ===")
print(f"📚 Total reading memories: {len(embeddings)}")
print(f"📖 Books analyzed: {len(unique_books)}")

# Find the most central (average) point
center_point = np.mean(embeddings_2d, axis=0)
distances_to_center = [np.linalg.norm(point - center_point) for point in embeddings_
most_central_idx = np.argmin(distances_to_center)

print(f"🎯 Most 'central' reading memory:")
print(f"   Book: {metadata[most_central_idx]['book_title']}")
print(f"   Type: {metadata[most_central_idx]['chunk_type']}")
print(f"   Text preview: {chunks[most_central_idx]['text'][:100]}...")

plt.show()

# Optional: Show distances between books
print("\n=== How Similar Are Your Books? ===")
book_centroids = {}
for book in unique_books:
    book_indices = [i for i, title in enumerate(book_titles) if title == book]
    book_centroids[book] = np.mean(embeddings_2d[book_indices], axis=0)

for i, book1 in enumerate(unique_books):
    for book2 in unique_books[i+1:]:
        distance = np.linalg.norm(book_centroids[book1] - book_centroids[book2])
        print(f"📏 {book1} ↔ {book2}: {distance:.2f}")
```

```
Loaded 31 reading memories
Books: ['The Return of the Native', 'Unknown', 'minor feelings, an Asian America
Reducing embeddings to 2D... (this might take a moment)

=== Quick Insights ===
📚 Total reading memories: 31
📖 Books analyzed: 4
🎯 Most 'central' reading memory:
   Book: Unknown
   Type: passage_reaction_pair
   Text preview: Book: Unknown by Unknown
Page: 257


Passage: "This man from Paris was now so disguised by his leath...
```






```
=== How Similar Are Your Books? ===
📏 The Return of the Native ↔ Unknown: 0.01
📏 The Return of the Native ↔ minor feelings, an Asian American reckoning.: 0.07
📏 The Return of the Native ↔ against the gods, the remarkable story of risk.: 0
📏 Unknown ↔ minor feelings, an Asian American reckoning.: 0.07
📏 Unknown ↔ against the gods, the remarkable story of risk.: 0.06
📏 minor feelings, an Asian American reckoning. ↔ against the gods, the remarkab
```

Meta Reasoning demo

> For full, see 'Saluda_Reasoning_Fragment - Colab.pdf'

```
================================================================================

=== SALUDA REASONING LAB ===
Enter situations to see how Saluda would reason about responding.
Try things like: 'Today was weird' or 'I'm feeling overwhelmed'
Type 'quit' to end

Situation to analyze: The job market is scary.

=== SALUDA'S REASONING PROCESS ===
Situation: "The job market is scary."
```