

Book Recommender using NLP

Carsten Lydeking

Zealand Business College

Oral Exam – AI and ML, 4th Semester

2025-06-11

Book Recommender using NLP

Book Recommender using NLP

Carsten Lydeking
Zealand Business College
Oral Exam – AI and ML, 4th Semester



- ▶ Increasing demand for privacy-preserving, local-first ML applications.
- ▶ Typical recommender systems rely on cloud APIs and user profiles.
- ▶ Goal: explore feasibility of a fully offline, content-based book recommender system.
- ▶ Research question:
How can a local ML model be used to recommend books based on natural language descriptions?

2025-06-11

Book Recommender using NLP

└ Motivation and Problem

Key Points (45 seconds):

- Start with privacy motivation - very current topic
- Emphasize "fully offline" - no cloud APIs
- Research question is central - pause after reading it
- If asked about motivation: mention GDPR, data breaches, user control

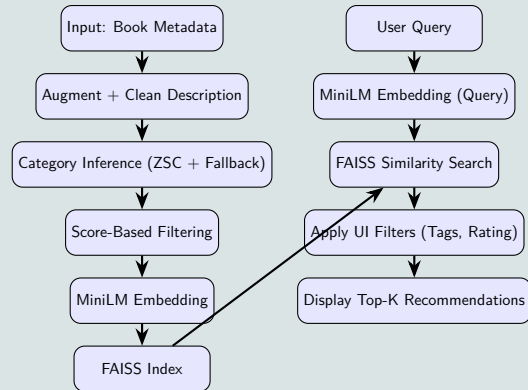
Potential Questions:

- "Why privacy focus?" → GDPR compliance, user trust, data breaches
- "What's wrong with cloud?" → Data collection, profiling, vendor lock-in



Modular, fully local processing pipeline:

- Data cleaning and augmentation
- Category inference via zero-shot classification + fallback
- Sentence embedding with MiniLM
- Fast vector similarity search with FAISS
- Offline UI built with Streamlit



2025-06-11

Book Recommender using NLP

Architecture Overview

Modular, fully local processing pipeline:

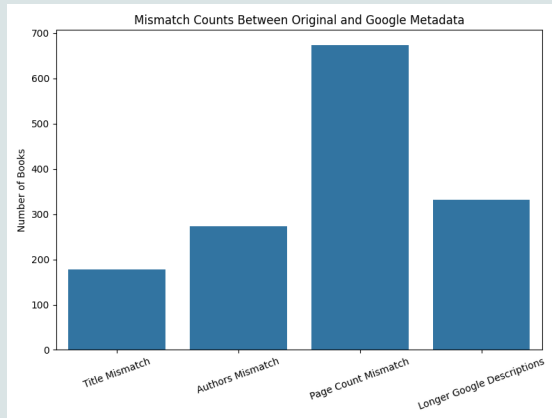
- Data cleaning and augmentation
- Category inference via zero-shot classification + fallback
- Sentence embedding with MiniLM
- Fast vector similarity search with FAISS
- Offline UI built with Streamlit





Original dataset ~ 6800 books

- ▶ Missing or inconsistent fields (authors, categories, descriptions)
- ▶ Very short or low-quality descriptions
- ▶ Category noise across sources
- ▶ OpenLibrary and Google Books API used to enrich metadata
- ▶ Rows with < 9 words in description removed
- ▶ Final dataset: 5160 high-confidence books



2025-06-11

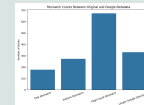
Book Recommender using NLP

└ Dataset Exploration

Dataset Exploration

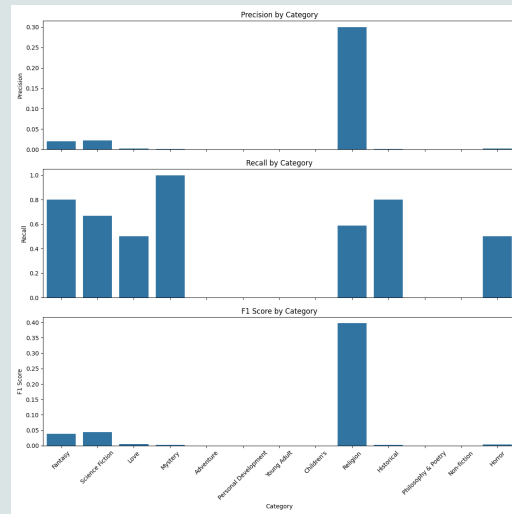
Original dataset ~ 6800 books

- ▶ Missing or inconsistent fields (authors, categories, descriptions)
- ▶ Very short or low-quality descriptions
- ▶ Category noise across sources
- ▶ OpenLibrary and Google Books API used to enrich metadata
- ▶ Rows with < 9 words in description removed
- ▶ Final dataset: 5160 high-confidence books





- Zero-shot classification with BART-MNLI
- 13 candidate categories defined
- Fallback keyword rules added for weak predictions
- Per-category metrics calculated:
 - Precision
 - Recall
 - F1-score
- Final filtering based on confidence thresholds:
 - `description_length` \geq 200 chars
 - `avg_score` \geq 0.2
 - `max_score` \geq 0.4



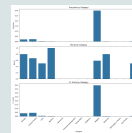
2025-06-11

Book Recommender using NLP

Category Inference

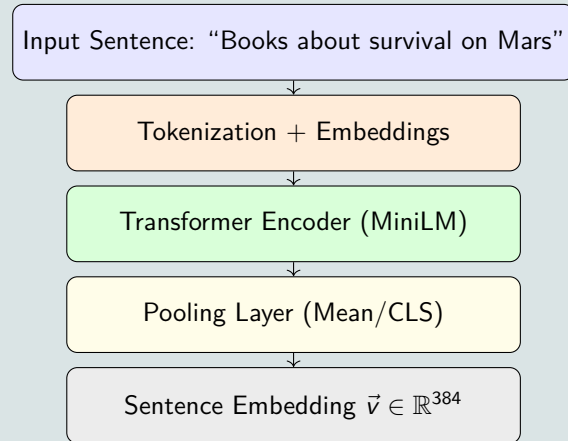
Category Inference

- Zero-shot classification with BART-MNLI
- 13 candidate categories defined
- Fallback keyword rules added for weak predictions
- Per-category metrics calculated:
 - Precision
 - Recall
 - F1-score
- Final filtering based on confidence thresholds:
 - `description_length` \geq 200 chars
 - `avg_score` \geq 0.2
 - `max_score` \geq 0.4





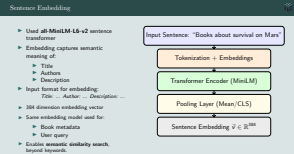
- ▶ Used **all-MiniLM-L6-v2** sentence transformer
- ▶ Embedding captures semantic meaning of:
 - ▶ Title
 - ▶ Authors
 - ▶ Description
- ▶ Input format for embedding:
Title: ... Author: ... Description: ...
- ▶ 384 dimension embedding vector
- ▶ Same embedding model used for:
 - ▶ Book metadata
 - ▶ User query
- ▶ Enables **semantic similarity search**, beyond keywords.



2025-06-11

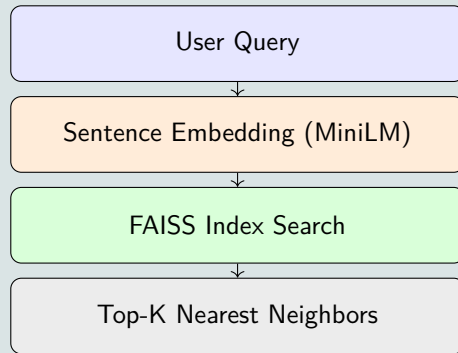
Book Recommender using NLP

└ Sentence Embedding





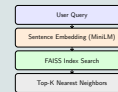
- Used **Facebook AI Similarity Search (FAISS)** library
- Performs **L2 distance** search in embedding space
- Index built with:
 - 5160 book embeddings ($\vec{v} \in \mathbb{R}^{384}$)
 - Exact search (IndexFlatL2)
- At runtime:
 - User query is embedded
 - Top-k nearest neighbors retrieved
 - Results shown in UI
- **Fully local**, fast search



2025-06-11

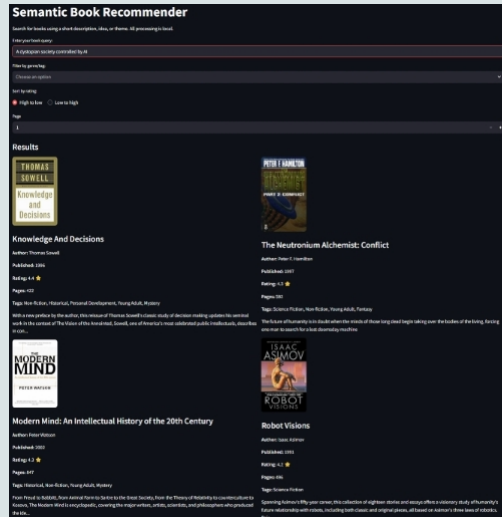
Vector Similarity Search with FAISS

- Used **Facebook AI Similarity Search (FAISS)** library
- Performs **L2 distance** search in embedding space
- Index built with:
 - 5160 book embeddings ($\vec{v} \in \mathbb{R}^{384}$)
 - Exact search (IndexFlatL2)
- At runtime:
 - User query is embedded
 - Top-k nearest neighbors retrieved
 - Results shown in UI
- **Fully local**, fast search





- ▶ Built with **Streamlit**
- ▶ **Fully offline** application:
 - ▶ No cloud calls
 - ▶ No tracking
 - ▶ No user profiles required
- ▶ Supports:
 - ▶ Natural language search
 - ▶ Filtering by category
 - ▶ Sorting by rating
 - ▶ Pagination of results
- ▶ Responsive UI:
 - ▶ Query time ≈ 200 ms
 - ▶ UI refresh ≈ 1 -2 sec (including images)



2025-06-11

Book Recommender using NLP

└ User Interface and Privacy

- ▶ Built with **Streamlit**
- ▶ **Fully offline** application:
 - ▶ No cloud calls
 - ▶ No tracking
 - ▶ No user profiles required
- ▶ Supports:
 - ▶ Natural language search
 - ▶ Filtering by category
 - ▶ Sorting by rating
 - ▶ Pagination of results
- ▶ Responsive UI:
 - ▶ Query time ≈ 200 ms
 - ▶ UI refresh ≈ 1 -2 sec (including images)





► **Strengths:**

- Fully local, privacy-preserving recommendation system
- Lightweight architecture
- Semantic search works well even for abstract queries

► **Limitations:**

- No personalization (no user profile or history)
- Dependent on description quality
- Dataset relatively small (~ 5160 books)
- No learning from user feedback

► **Possible improvements:**

- Larger, more diverse dataset
- Explore better embedding models (MPNet, SBERT)
- Implement re-ranking layer

"Even compact transformer models provide robust performance on consumer hardware."

2025-06-11

Book Recommender using NLP

└ Reflections and Limitations

Reflections and Limitations

► **Strengths:**

- Fully local, privacy-preserving recommendation system
- Lightweight architecture
- Semantic search works well even for abstract queries

► **Limitations:**

- No personalization (no user profile or history)
- Dependent on description quality
- Dataset relatively small (~ 5160 books)
- No learning from user feedback

► **Possible improvements:**

- Larger, more diverse dataset
- Explore better embedding models (MPNet, SBERT)
- Implement re-ranking layer

"Even compact transformer models provide robust performance on consumer hardware."



► Research question:

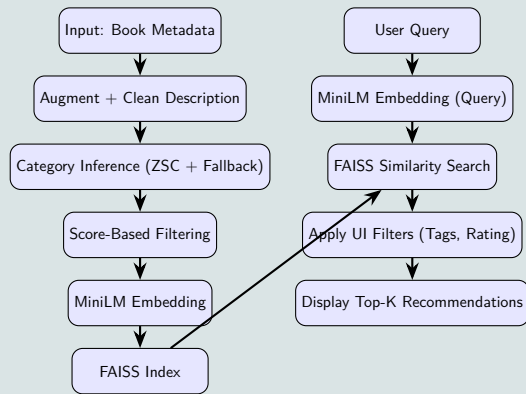
How can a local ML model be used to recommend books based on natural language descriptions?

► Summary:

- Developed a fully local recommendation system
- Used semantic embeddings and vector similarity search
- No cloud, no tracking — privacy-first
- User interface for natural language queries

► Key takeaway:

- "Local-first ML applications are practical and effective for semantic recommendation tasks."

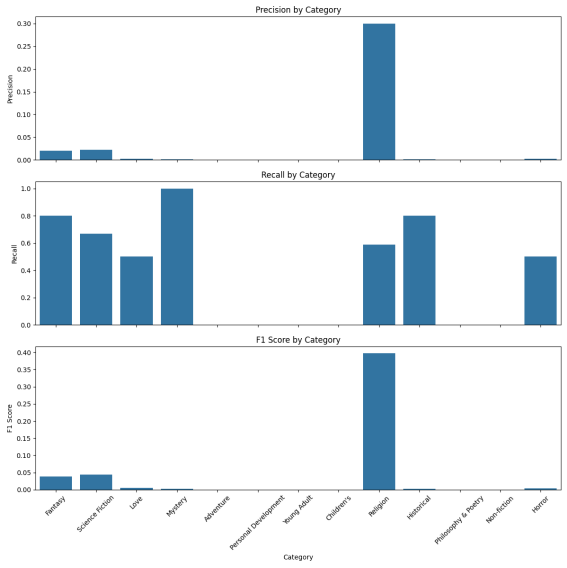


2025-06-11

Conclusion

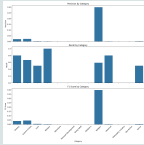
Conclusion

- **Research question:**
How can a local ML model be used to recommend books based on natural language descriptions?
- **Summary:**
 - Developed a fully local recommendation system
 - Used semantic embeddings and vector similarity search
 - No cloud, no tracking — privacy-first
 - User interface for natural language queries
- **Key takeaway:**
 - "Local-first ML applications are practical and effective for semantic recommendation tasks."



2025-06-11

Backup: Per-Category Metrics





- ▶ Fallback used when zero-shot model confidence was low
- ▶ Example keywords:
 - ▶ **Fantasy**: magic, wizard, dragon
 - ▶ **Science Fiction**: space, AI, dystopia
 - ▶ **Love**: romance, passion, relationship
 - ▶ **Mystery**: detective, clue, crime

2025-06-11

Book Recommender using NLP

└ Backup: Fallback Keywords

- ▶ Fallback used when zero-shot model confidence was low
- ▶ Example keywords:
 - ▶ **Fantasy**: magic, wizard, dragon
 - ▶ **Science Fiction**: space, AI, dystopia
 - ▶ **Love**: romance, passion, relationship
 - ▶ **Mystery**: detective, clue, crime



- Embedding time per book: ≈ 2 ms (batch embedding)
- Query embedding: ≈ 50 -200 ms
- FAISS search: < 10 ms
- UI render time: ≈ 1 -2 seconds (including image loading)
- All processing fully local on consumer-grade laptop

2025-06-11

Book Recommender using NLP

└ Backup: Performance

- Embedding time per book: ≈ 2 ms (batch embedding)
- Query embedding: ≈ 50 -200 ms
- FAISS search: < 10 ms
- UI render time: ≈ 1 -2 seconds (including image loading)
- All processing fully local on consumer-grade laptop