Book Recommender using NLP

Carsten Lydeking

Zealand Academy of Technologies and Business

AI and ML, 4th Semester

[OPENING (30 seconds):]

- "Good morning, I'm Carsten Lydeking"
- "Today I'll present my book recommender system using NLP"
- "This is for my AI and ML oral exam, 4th semester"
- "I'll speak for about 8-9 minutes, then happy to take questions"

[BODY LANGUAGE:]

- Stand confidently, make eye contact with all examiners
- Hands visible, relaxed posture
- Speak clearly and at moderate pace

[TRANSITION:] "Let me start by explaining what motivated this project..."

Motivation and Problem

- **Growing privacy and cost concerns** with cloud-based systems
- Traditional systems require **user profiles and data collection**
- **Goal:** Build a fully offline, content-based book recommender
- **Research Question:**

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

- **Key Requirements:** No external APIs, no user tracking, semantic understanding

[TIMING: 45 seconds - KEY TALKING POINTS:]

- "Privacy and independence concerns are growing - GDPR, data breaches, user awareness"
- "Traditional recommenders like Amazon collect massive user data"
- "My goal: prove you can do semantic recommendations locally"
- PAUSE after research question - let it sink in
- "Three key requirements guide everything I built"

[DEFINE THESE TERMS:]

- Content-based: Recommends based on item features, not user behavior
- Semantic understanding: Goes beyond keywords to understand meaning
- Local/Offline: Runs entirely on user's device, no internet needed

[POTENTIAL QUESTIONS:]

- *"Why privacy and independence focus?"* → GDPR compliance, user control, vendor independence, cost control
- *"What's wrong with cloud APIs?"* → Vendor lock-in, data collection, cost, dependency on enterprise software
- *"Content vs collaborative filtering?"* → Content uses item features, collaborative uses user behavior patterns

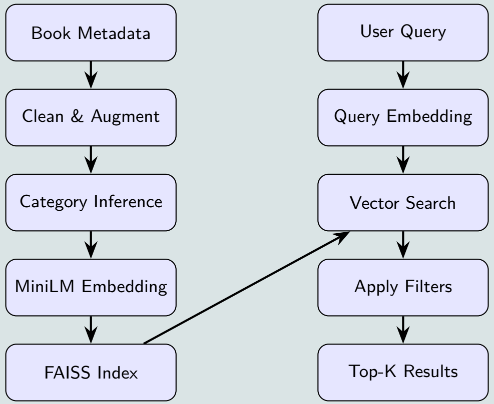
[TRANSITION:]

- "Let me show you the architecture I designed to achieve this..."

Core Components:

- **Data Pipeline:** Clean & augment book metadata
- **Category Inference:** Zero-shot classification + fallback rules
- **Semantic Embedding:** `MiniLM` sentence transformers
- **Vector Search:** `FAISS` similarity matching
- **Local UI:** `Streamlit` interface

Key Innovation: Fully local semantic search without cloud dependencies



[TIMING: 1 minute - DIAGRAM WALKTHROUGH:]

- "Left side processes book data offline" (point to diagram)
- "Right side handles user queries in real-time"
- "Data flows from top to bottom, then connects for search"
- "Everything happens locally - no network calls"

[DEFINE KEY TERMS:]

- Zero-shot classification: "Model classifies without training examples - understands categories from descriptions alone"
- MiniLM: "Lightweight transformer model - like GPT but smaller, designed for understanding meaning"
- FAISS: "Facebook's library for fast similarity search - finds nearest neighbors in high-dimensional space"
- Semantic embedding: "Converting text to numbers that represent meaning - similar concepts get similar numbers"

[KEY INNOVATION EMPHASIS:]

- "The innovation is making semantic search work locally and independently"
- "Usually requires cloud APIs like OpenAI or Google"
- "I prove you can do it on consumer hardware without dependencies"

[POTENTIAL QUESTIONS:]

- *"What's a transformer?"* → "Neural network architecture very good at understanding language context"
- *"Why modular design?"* → "Easy to swap components, test approaches, maintain independence"
- *"How does semantic differ from keyword search?"* → "Keywords match exact words, semantic understands concepts"

[TRANSITION:]

- "Let me dive into the data challenges I had to solve first..."

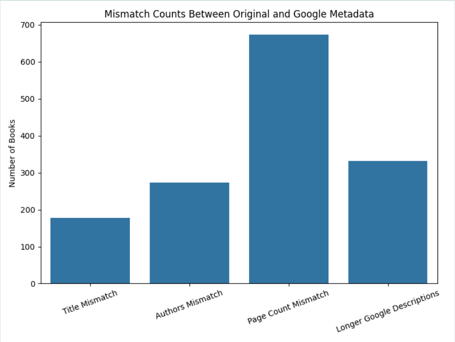
Dataset Exploration & Processing

Data Quality Challenges:

- Started with **6,800 books** from multiple sources
- **Issues found:**
  - Missing author/category information
  - Very short descriptions (< 9 words)
  - Inconsistent categorization across sources

Data Engineering Solutions:

- **API enrichment:** OpenLibrary & Google Books
- **Quality filtering:** Remove inadequate descriptions
- **Final dataset:** 5,160 high-quality books



Data inconsistencies across sources required systematic cleaning and validation

[TIMING: 45 seconds - DATA ENGINEERING STORY:]

- "Real-world data is always messy - this was no exception"
- "Started with 6,800 books but quality varied dramatically"
- "Some books had 2-word descriptions like 'Great book!' - useless for NLP"
- "Categories were inconsistent - same book labeled 'Fiction' and 'Novel' and 'Literature'"

[EXPLAIN THE CHART (point to it):]

- "This shows mismatches between original data and Google Books"
- "Page count mismatches were highest - shows data quality issues"
- "Had to decide: fix manually or filter automatically"
- "Chose filtering for scalability"

[TECHNICAL DECISIONS:]

- "9-word minimum because shorter descriptions don't contain enough semantic information"
- "API enrichment instead of manual correction - more scalable"
- "25% data loss acceptable for quality gain"

[POTENTIAL QUESTIONS:]

- *"Why not keep all data?"* → "Garbage in, garbage out - quality over quantity for NLP"
- *"5,160 seems small?"* → "Proof-of-concept scale, approach scales to millions"
- *"How handle missing authors?"* → "API enrichment first, then filtering if still missing"
- *"Other data sources?"* → "Could add Goodreads, library catalogs, but these were sufficient"

[TRANSITION:]

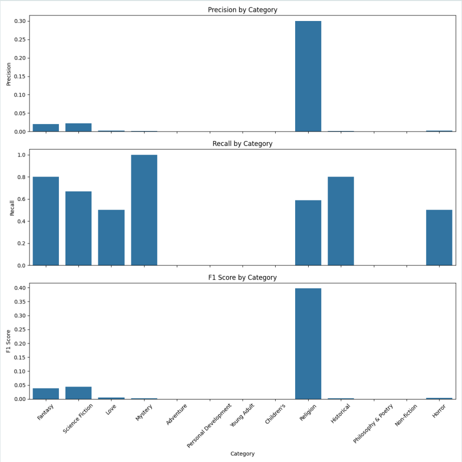
- "With clean data, I could focus on the NLP challenge of categorization..."

Two-Tier Classification Approach:

- **Primary:** Zero-shot classification with `BART-MNLI`
  - No training data required
  - 13 predefined book categories
  - Confidence scoring for predictions
- **Fallback:** Rule-based keyword matching
  - When confidence $<$ threshold
  - Genre-specific keyword patterns

Quality Control:

- Description length $\geq$ 200 chars
- Average confidence $\geq$ 0.2
- Maximum confidence $\geq$ 0.4



Results focus on high-confidence predictions rather than perfect recall across all categories

[TIMING: 1 minute - WHY TWO-TIER APPROACH:]

- "Zero-shot is powerful but not perfect"
- "Sometimes misses obvious genre indicators"
- "Fallback catches cases like book description containing 'wizard' but AI missed fantasy"
- "Hybrid approach gets best of both worlds"

[EXPLAIN ZERO-SHOT (key concept):]

- "Zero-shot means no training examples needed"
- "Just ask BART: 'Is this book description about fantasy?'"
- "It understands the question and gives confidence score"

[ADDRESS THE CHART HONESTLY:]

- "Yes, precision is low for some categories That's expected - I'm prioritizing high-confidence predictions"
- "Religious category shows best performance - clear vocabulary indicators"
- "This is proof-of-concept, not production system"

[QUALITY CONTROL RATIONALE:]

- "200 chars minimum - need enough text for semantic analysis"
- "Confidence thresholds prevent low-quality predictions"

[POTENTIAL QUESTIONS:]

- *"Why BART-MNLI specifically?"* $\rightarrow$ "Pre-trained for natural language inference - exactly what we need for 'is this book about X?'"
- *"How choose the 13 categories?"* $\rightarrow$ "Common book genres, could be expanded based on dataset"
- *"What if zero-shot and fallback disagree?"* $\rightarrow$ "Take highest confidence, with preference for zero-shot"
- *"Could you improve these metrics?"* $\rightarrow$ "Yes - larger training set, domain-specific fine-tuning, better thresholds"

[TRANSITION:]

- "With categories assigned, the core challenge was semantic embedding..."

Sentence Embedding with `MiniLM`:

- **Model:** `all-MiniLM-L6-v2`
- **Input format:** *"Title: ... Author: ... Description: ..."*
- **Output:** 384-dimensional vectors
- **Advantage:** Semantic similarity beyond keywords

Vector Search with `FAISS`:

- **Index:** 5,160 book embeddings
- **Search:** L2 distance (exact search)
- **Performance:** $<$ 10ms query time
- **Local:** No external dependencies

Why `MiniLM` Specifically:

- Balance of performance vs size
- Small enough to run locally in real-time
- But powerful enough for semantic understanding

Query: "space exploration novels"

↓

MiniLM Transformer

↓

Vector: $\vec{v} \in \mathbb{R}^{384}$

↓

FAISS L2 Distance Search

↓

Top-K Similar Books

End-to-end semantic search in $<$ 200ms

[TIMING: 1.5 min Embeddings - CORE TECHNICAL SLIDE]

- "Embeddings convert text to numbers that capture meaning - 384 dimensions means each book becomes a point in 384D space"
- "Books with similar meanings cluster together in this space -Like a map where distance represents semantic similarity"

[CONCRETE EXAMPLE (use the diagram):]

- "User types 'space exploration novels' - MiniLM converts this to 384 numbers"
- "FAISS finds books whose embeddings are closest"
- "Might find 'Mars colonization story', 'astronaut memoir' - no exact keyword matches needed!"

[PERFORMANCE EMPHASIS:]

- "Sub-200ms total - feels instant to users"
- "Most time in embedding query, search is nearly instant"
- "Scales well - current dataset tiny compared to FAISS capabilities"

[TECHNICAL DEPTH (if asked):]

- "Could use cosine similarity but L2 works well for normalized embeddings"
- "FAISS uses optimized algorithms for billion-scale search"
- "IndexFlatL2 = exact search, could use approximate for speed"

[POTENTIAL QUESTIONS:]

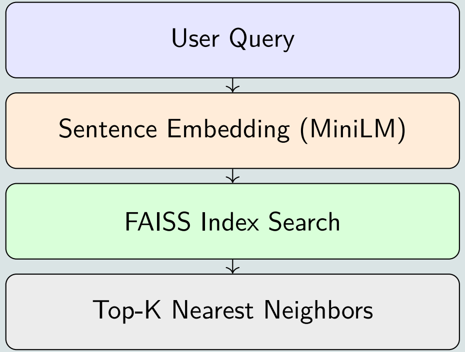
- *"Why not larger models like BERT?"* → "Size vs performance tradeoff - need to run locally"
- *"How does semantic similarity work?"* → "Model trained to put similar meanings close together in vector space"
- *"What if user query very different from training?"* → "May not work well - limitation of current approach"
- *"Could you use approximate search?"* → "Yes, FAISS has IVF, LSH options for speed vs accuracy tradeoff"

[TRANSITION:]

- "Let me show you how this looks in the actual user interface..."

Vector Similarity Search with FAISS

- Used **Facebook AI Similarity Search (FAISS)** library which performs L2 (Euclidean) 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
- Example Queries
  - "Instead of 'fantasy dragons', user can type 'books about magical creatures'"
  - "'philosophical science fiction' finds books exploring deep questions"
  - "'unreliable narrator mystery' - semantic understanding of literary techniques"



[TIMING: 1 minute - DEMO THE INTERFACE (point to screenshot):]

- "Simple, clean interface built with Streamlit"
- "User types natural language - no complex syntax"
- "Results appear instantly with covers and descriptions"
- "Can filter by genre, sort by rating"

[PRIVACY AND INDEPENDENCE EMPHASIS:]

- "No data ever leaves the user's device - No tracking cookies, no user profiles, no analytics"
- "User has complete control - can run offline"
- "Contrast with Amazon, Goodreads - they track everything"

[TECHNICAL IMPLEMENTATION:]

- "Streamlit chosen for rapid prototyping"
- "Could be ported to web app, mobile app, desktop app"
- "All processing happens in Python backend"
- "UI just displays results, no smart client needed"

[PERFORMANCE DETAILS:]

- "200ms feels instant to users"
- "Most delay is loading book cover images"
- "Could optimize with image caching, lazy loading"
- "Runs smoothly on 4GB RAM laptop"

[POTENTIAL QUESTIONS:]

- *"Could this scale to millions of books?"* → "Yes, FAISS designed for billion-scale, just need more RAM"
- *"What about mobile deployment?"* → "Would need model quantization, smaller embedding dimension"
- *"How update book database?"* → "Currently manual, could automate with book APIs"
- *"Could users add their own books?"* → "Yes, just run embedding pipeline on new books"

[TRANSITION:]

- "To reflect on the project, we can consider the following aspects..."

Implementation Strengths:

- **Privacy-preserving** by design
- **Semantic understanding** beyond keyword matching
- **Lightweight** - runs on consumer hardware
- **Modular architecture** for easy extension

Current Limitations:

- **No personalization** - stateless by design
- **Dataset scope** - 5,160 books vs. commercial scale
- **Cold start problem** for new books
- **No feedback learning** - static recommendations

Edge AI:

- Edge AI is popular term for local-first ML
- Growing trend in mobile, home, IoT, privacy applications
- The project fits this paradigm perfectly

Future Research Directions:

- **Hybrid approach:** Combine content-based with collaborative filtering
- **Better embeddings:** Experiment with domain-specific models
- **Privacy-preserving personalization:** Local user preference learning
- **Multi-modal features:** Include cover images, genre embeddings

"Demonstrates that local-first ML - or using a more popular term - edge AI, can provide meaningful semantic recommendations without compromising user privacy or requiring cloud infrastructure."

[TIMING: 1 minute - HONEST SELF-ASSESSMENT:]

- "This is proof-of-concept, not production system"
- "Goal was to demonstrate feasibility, not perfection - Garbage In, Garbage Out"

[LIMITATIONS:]

- "No personalization - everyone gets same results for same query"
- "Dataset small compared to Amazon's millions of books"
- "New books need manual addition and embedding - batch trained"

[FUTURE DIRECTIONS:]

- "Hybrid: Add collaborative filtering while preserving privacy and independence"
- "Better embeddings: Fine-tune MiniLM on book-specific corpus"
- "Local personalization: Session-based learning without tracking"
- "Multi-modal: Computer vision on book covers for genre signals"

[POTENTIAL QUESTIONS:]

- *"How would you add personalization?"* → "Local preference vectors, federated learning, session-based adaptation"
- *"Could this work for other domains?"* → "Yes - movies, music, research papers, any content with descriptions"
- *"What's the biggest technical limitation?"* → "Embedding quality for domain-specific queries"
- *"How evaluate recommendation quality?"* → "User studies, semantic similarity benchmarks, domain expert evaluation"

[TRANSITION:]

- "Let me conclude by answering the original research question..."

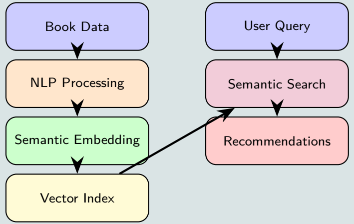
Research Question Answered:

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

Solution Implemented:

- **Semantic embeddings** with MiniLM transformers
- **Vector similarity search** using FAISS
- **Zero-shot classification** for categorization
- **Privacy-first design** - fully local processing
- **Key Contributions:** *Proof-of-concept that modern NLP enables practical, independent, privacy-preserving recommendation systems*

System Architecture:



Impact & Applications:

- Educational tool for ML/Edge AI concepts
- Foundation for local-first recommendation systems
- Demonstrates transformer accessibility on consumer hardware

[TIMING: 45 seconds - DIRECTLY ANSWER RESEARCH QUESTION:]

- "The answer: semantic embeddings + vector search + zero-shot classification"
- "All running locally without cloud dependencies - Proven to work on real data with real performance"

[KEY CONTRIBUTIONS (what's proven):]

- "Modern NLP makes privacy-preserving recommendations feasible"
- "Transformer models are accessible to individual developers"
- "Local-first doesn't mean sacrificing functionality"
- "Edge AI is practical for semantic tasks"

[BROADER IMPACT:]

- "Educational value - shows students what's possible"
- "Template for other local-first applications"
- "Contribution to privacy-preserving ML research"

[CONFIDENT CLOSING:]

- "This project demonstrates that the future of recommendations doesn't require surrendering privacy"
- "With modern NLP tools, we can have both semantic understanding AND user control"
- "Thank you - I'm happy to answer any questions"

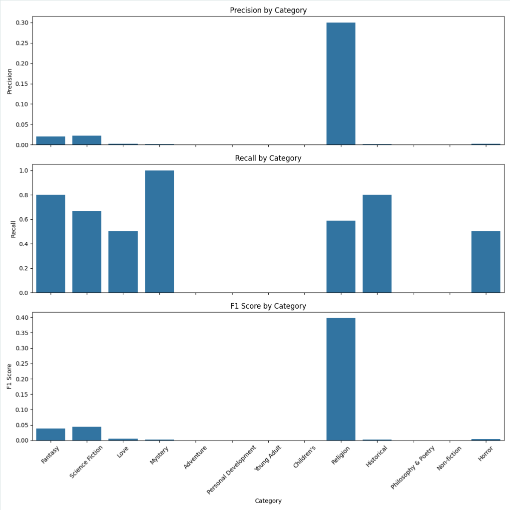
[POTENTIAL IMMEDIATE FOLLOW-UPS:]

- *"Can you show us the actual system?"* → "Happy to demo if we have time"
- *"What was the biggest challenge?"* → "Data quality and choosing the right embedding model"
- *"How long did this take?"* → "X weeks for implementation, focus on learning modern NLP tools"
- *"What would you do differently?"* → "Start with better dataset, experiment with domain-specific embeddings"

Thank you for your attention!

Questions?

Backup: Per-Category Metrics



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

Backup: Performance

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

Example Query 1:

"Books about artificial intelligence and ethics"

Top Results:

- "The Alignment Problem" - Brian Christian
- "Life 3.0" - Max Tegmark
- "Weapons of Math Destruction" - Cathy O'Neil

Example Query 2:

"mystery novels with unreliable narrators"

Top Results:

- "Gone Girl" - Gillian Flynn
- "The Girl on the Train" - Paula Hawkins
- "In the Woods" - Tana French

What This Demonstrates:

- **Semantic understanding** beyond keywords
- **Abstract concept matching** (ethics, unreliable narrators)
- **Cross-genre discovery** potential

Evaluation Challenges:

- No ground truth for "perfect" recommendations
- Subjective nature of book preferences
- **Solution:** Focus on semantic relevance rather than prediction accuracy

Discussion Starters:

- How could one evaluate recommendation quality
- Books with sparse descriptions
- Could this approach work for other domains?