Simple book recommender

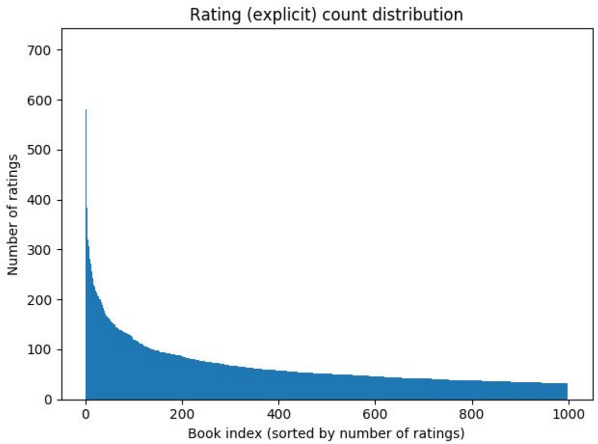
Tomáš Novák
tomas2211@post.cz
github.com/tomas2211/book_recommender

Outline

- Dataset + Task
- Approaches
  - Simple baseline approach
  - Graph-based approaches - close vertices, node2vec
  - K nearest neighbours
- Evaluation
  - NDCG
  - Qualitative evaluation
- Deployment

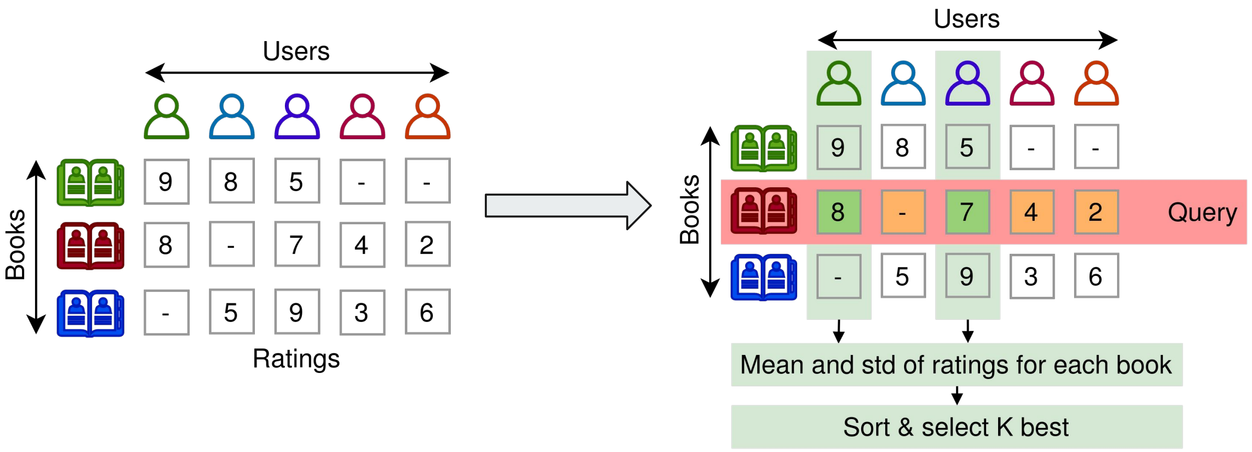
Dataset + Task

- Data: User ratings
  - Columns - user-id, book-id, rating
  - Ratings - Implicit (0), explicit (1-10)
    - Considering only explicit ratings
  - Many books with small number of ratings
    - Considering only books with >10 ratings
    - → 4963 books
- Task
  - Input: Query (book ISBN)
  - Output: Top K recommended books



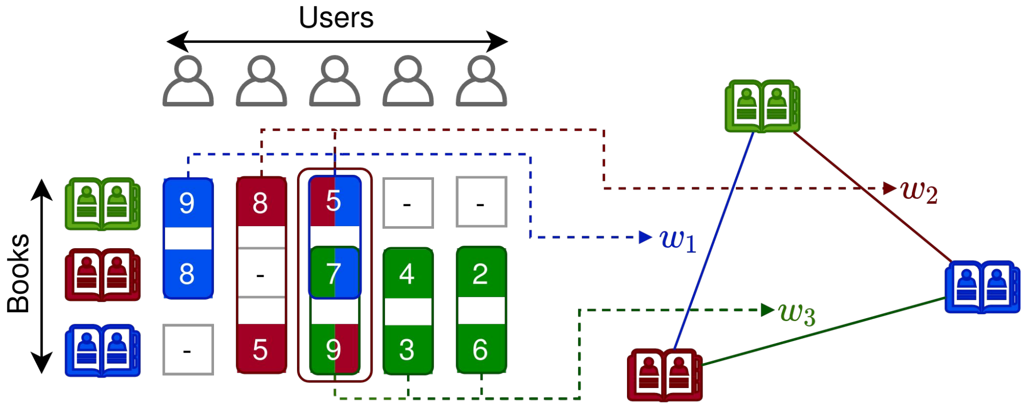
Baseline approach

- Idea: model conditional prob. distributions of ratings for each book
- **P(*rating of book* | *liked query*) ≈ N(μ,σ²)**
- Liked the query: rating > 5 (fixed threshold)
- Maximize the minimum expected rating on some level of confidence i.e. **max(μ - S*σ)**



Graph-based approach

- Idea: model relations between books in a graph
  (vertices = books, edges = relations, weighted by distance/closeness)
- Possibility to explore deeper relations, find niches
- Weights calculated by considering each pair of user's ratings



Graph-based approach

- Edge weights
  - User's u rating of book k: $r_u(k)$
  - **Co-rating** of books k and l: $r_u(k)r_u(l)$
  - Mean and std over users u that rated k and l, normalized:

$$\mu(k,l) = \frac{1}{N} \sum_{u:r_u(k)>0, r_u(l)>0} \frac{r_u(k)r_u(l)}{100}$$

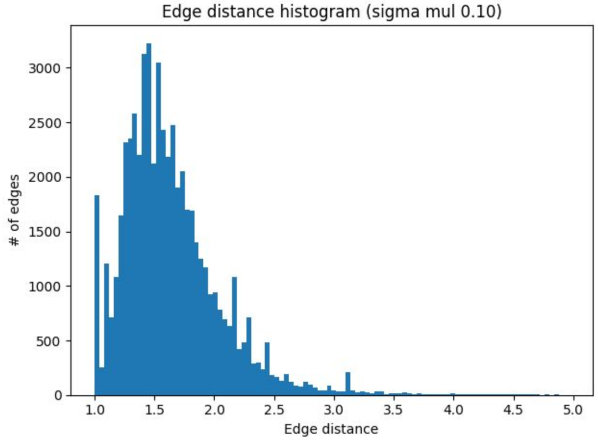
$$\sigma(k,l) = \sqrt{\frac{1}{N} \sum_{u;r_u(k)>0, r_u(l)>0} \left(\frac{r_u(k)r_u(l)}{100} - \mu(k,l)\right)^2}$$

(+ disregard edges with less than 3 co-ratings, ...)

Close vertices - Graph-based approach

- Search around query vertex - limit depth
- Sort other vertices by distance (path length) to query vertex
- Needs “distance measure” - proportional to dissimilarity
    - Inverse mean co-rating
    - “Standard deviation trick” with parameter S

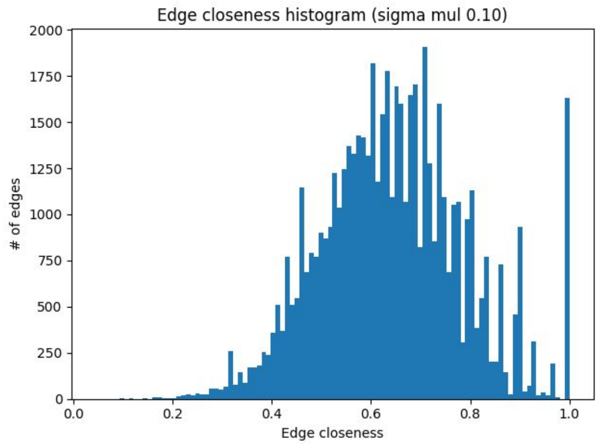
$$d(k,l) = \frac{1}{(\mu(k,l) - S\sigma(k,l))}$$



Node2Vec - Graph-based approach

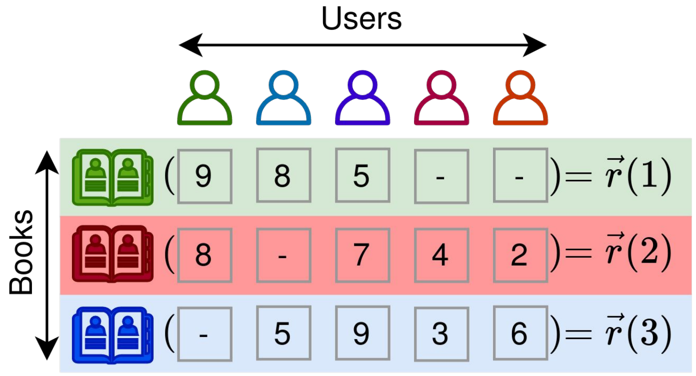
- Create vector representations (embeddings) for nodes in graph
- Based on Word2Vec (sequences of nodes by random walks)
- Compare vectors with query embedding (cosine distance)
- Edges need “closeness measure”
  - Mean co-rating normalized to range 0 - 1
  - “Standard deviation trick”

$$c(k,l) = (\mu(k,l) - S\sigma(k,l))$$



k-nearest neighbours

- Conventional approach
- Represent books as vectors of user ratings
- Cosine similarity with query book
  - Similar to co-ratings, but normalized differently
- Measures similarity instead of probability of liking a book
  - If one users dislikes two books - makes them more similar
- Potential problem on less-rated books - cannot step ‘over a user’



$$sim(k,l) = \frac{\sum_u r_u(k) r_u(l)}{\sqrt{\sum_u r_u^2(k)}\sqrt{\sum_u r_u^2(l)}}$$

Quantitative evaluation

- Randomly selected test users (excluded from training)
  - 40 users with more than 50 explicit ratings
- Query each ‘liked’ book of test user (>5 rating)
- Response - top 10 books
- Compute NDCG on response
  - Sum user ratings of responses, weighted by place
  - Unrated books - rating = 0
  - Normalize by maximum possible score for the user

$$\frac{1}{\log_2(place + 1)}$$

Quantitative evaluation

- Overall, low scores
  - Rating sparsity?
  - Single liked book may not bring much information about the user
  - Maximal attainable NDCG?
- Better than a random selection
  - Probability of randomly hitting a rated book is 1.6%
- Graph - close nodes
  - Better selection of rated nodes
  - Worse cumulative gain - distribution of distances to narrow?

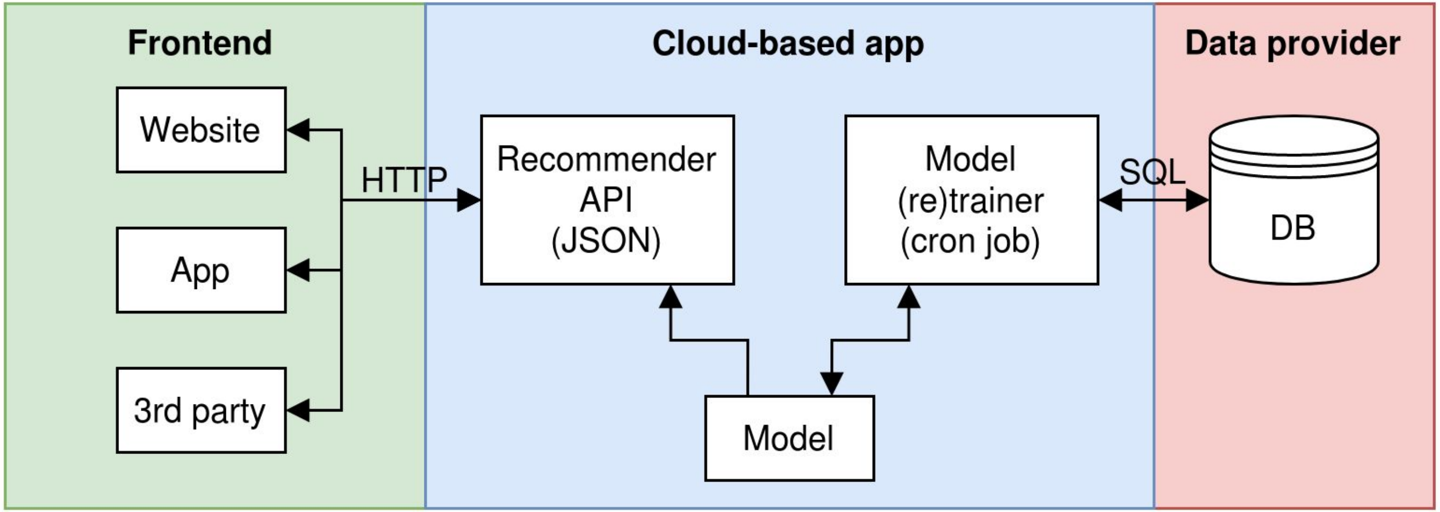
Approach	NDCG	Rated books in responses [%]
Baseline	0.092	10.7
Graph - close vertices	0.097	16.6
Graph - node2vec	0.102	12.1
kNN	0.118	12.6

Qualitative evaluation

- Mostly relevant or at least not irritating suggestions, but with exceptions (selections from top 5)

Approach	J. R. R. Tolkien: The Hobbit	Chaim Potok: The Chosen	Douglas Adams: Hitchhiker's Guide to the Galaxy
Baseline	J. K. Rowling: Harry Potter and the Sorcerer's Stone	Margaret Atwood: The Handmaid's Tale [dystopia]	Jon Krakauer: Into Thin Air : A Personal Account of the Mt. Everest Disaster [non-fiction]
Graph - close vertices	J. K. Rowling: Harry Potter and the Sorcerer's Stone	John Le Carre: The Tailor of Panama [spy novel]	Dan Brown: The Da Vinci Code
Graph - node2vec	Helen Fielding: Das Tagebuch Der Bridget Jones	John Le Carre: The Tailor of Panama [spy novel]	Douglas Adams: The Restaurant at the End of the Universe
kNN	Martha Sacks: Menopaws: The Silent Meow	Chaim Potok: My Name Is Asher Lev	Douglas Adams: The Restaurant at the End of the Universe.

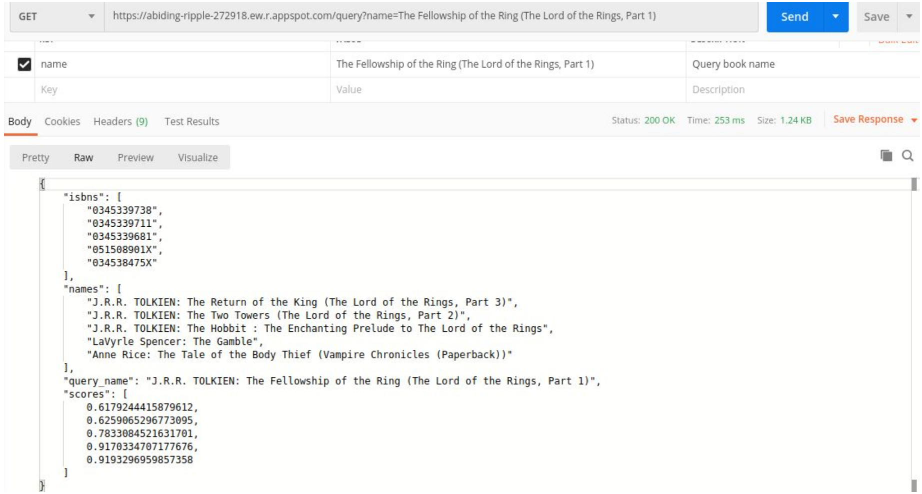
Productionalization



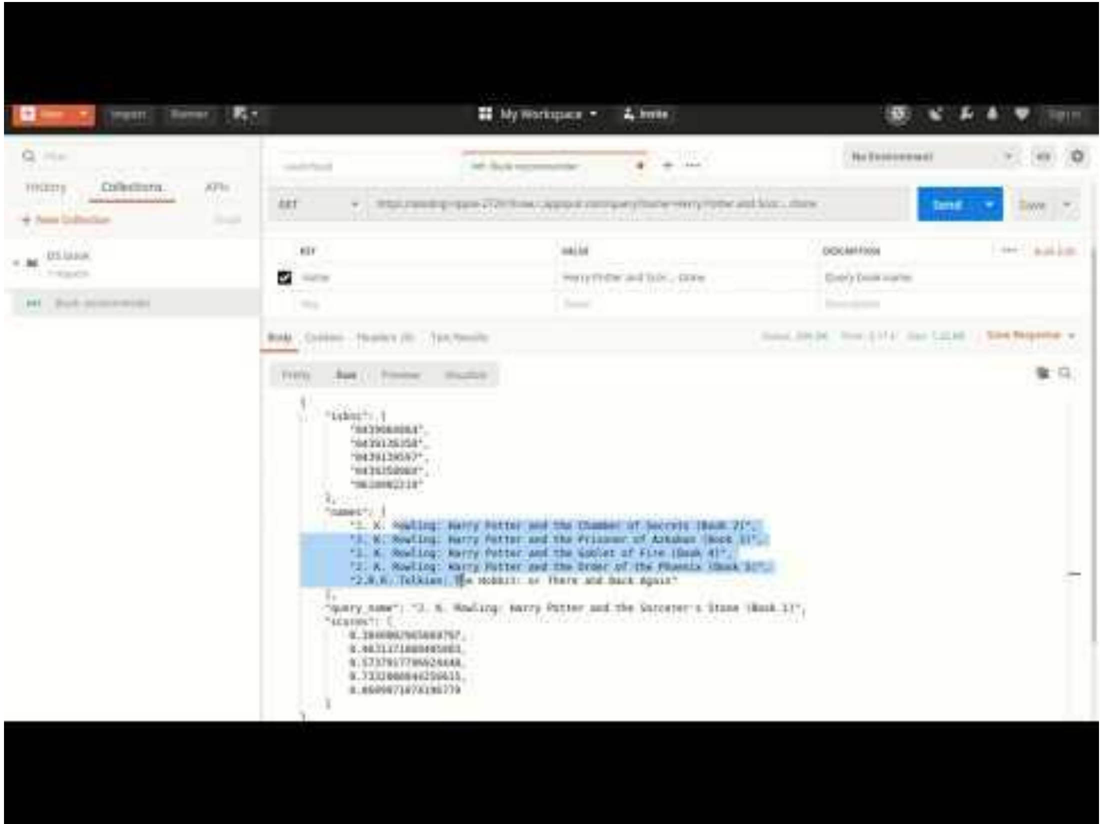
Google Cloud - App Engine

- JSON API in Flask, static model (kNN) - see *gcp_app* folder in repo
- Matches name against DB & evaluates the model

[https://abiding-ripple-272918.ew.r.appspot.com/query?name=[book-name]](https://abiding-ripple-272918.ew.r.appspot.com/query?name=[book-name])



Demo



Conclusions

- Designed several approaches to book recommendation
  - Baseline
  - Graph-based - close vertices, node2vec
  - kNN
- Evaluation - NDCG, qualitative
- Productionalization - design
- Prototype deployment to GCP - App Engine

Technologies: Python, Pandas, Numpy, Scikit-learn, NetworkX, Flask
Google cloud platform - App Engine