Book Recommendation System based on Amazon Book Reviews Dataset

Simple & Scalable Baseline

CS-GY 6513: Big Data (Fall 2024) - Project Presentation

Team: Daoyu Li, Dongzhe Fan, Penghao Weng, Siqi Wan, Xiaochen Lu

December 17, 2024

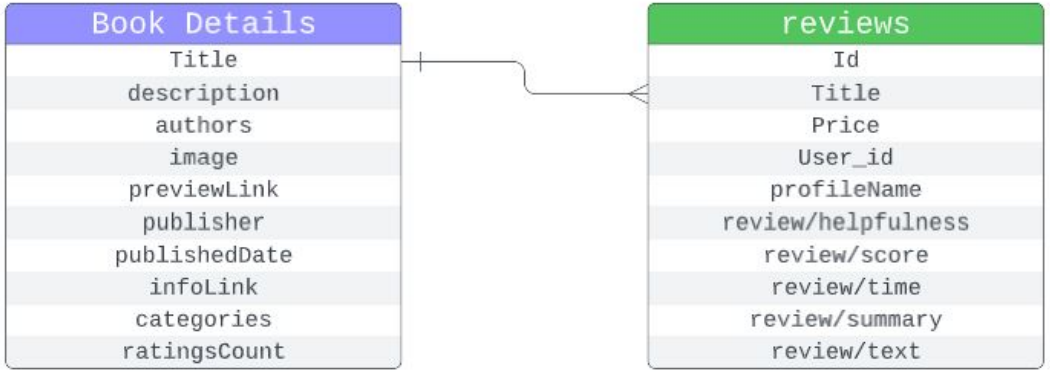
Outline

- Dataset Overview - Amazon Book Reviews
- Methodology
- Data Pre-processing
    - Word2Vec
    - Cosine Similarity Missing Label Prediction
- Collaborative Filtering
    - ALS using User Ratings
    - Auxiliary Content-based Filtering: Category Labels

Datasets Overview

Dataset Overview

- **Dataset Info**: Amazon Book Reviews
  - **Two Tables**
    - Book Details: detailed information about each book – description, category, etc.
    - Review: user reviews on books, including rating, review summary, etc., connected to the Book Details table via foreign key "book_id"
  - **Source**: https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews



- **Dataset Size**: Amazon Book Reviews
  - **Book Details**: 181.35MB, contains 212,404 unique books
  - **Review**: 2.86GB, contains 3,000,000 reviews

Dataset JSON Format

Book Details:


```json
{
    "title": "The Church of Christ: A Biblical Ecclesiology for Today",
    "description": "In The Church of Christ: A Biblical Ecclesiology for Today, respe
    "authors": ["Everett Ferguson"],
    "image": "http://books.google.com/books/content?id=kVqRaiPlx88C&printsec=frontcov
    "previewLink": "http://books.google.nl/books?id=kVqRaiPlx88C&printsec=frontcover&
    "publisher": "Wm. B. Eerdmans Publishing",
    "publishedDate": "1996",
    "infoLink": "http://books.google.nl/books?id=kVqRaiPlx88C&dq=The+Church+of+Christ
    "categories": ["Religion"],
    "ratingsCount": 5.0
}
```


Ratings:


```json
{
    "Id": "0802841899",
    "Title": "The Church of Christ: A Biblical Ecclesiology for Today",
    "Price": 25.97,
    "User_id": "ARI272XF8TOL4",
    "profileName": "Christopher J. Bray",
    "review": {
        "helpfulness": "74/81",
        "score": 5.0,
        "time": 955411200,
        "summary": "Ecclesiological Milestone",
        "text": "With the publication of Everett Ferguson's book on ecclesiology, another milestone has been reached in the
    }
}
```

Methodology

Methodology

- **Steps**:
  - **Data Preprocessing**: *Word2Vec* embeddings + *Cosine Similarity* to fill missing category labels.
  - **Collaborative Filtering**: *ALS* model training for book recommendations (based on users' ratings on books).
  - **Auxiliary Content-based Filtering**: Combine *ALS* results with content-based filtering using book Category labels.

- **Technologies**:
  - **PySpark, PyTorch (Word2Vec), Spark ML, and Apache Spark**

Missing Category Label Prediction

Data Pre-processing

- **Problem**:
  - **19% of book categories** in the dataset are missing.
  - Missing categories reduce the quality of recommendations, as they hinder content-based filtering.
- **Solution**: To impute the missing categories:
  - **Word2Vec Embeddings**: Convert raw text (book titles and descriptions) into embeddings that capture semantic meaning.
  - **Cosine Similarity**: Measure the similarity between books based on their embeddings.
  - **Assign Categories**: For each book with a missing category:
    - Identify the **top-10 most similar books**.
    - Choose the category with the **highest frequency** among these books.

Implementation

- **Step 1**: Preprocess Book Titles and Descriptions
  - Before training Word2Vec:
    - **Clean and tokenize** book titles and descriptions.
    - Ensure consistent text formatting.

```
data = spark.read.csv(BOOKS_DATA_FILE_PATH, header=True, schema=RAW_DATA_SCHEMA)
for column in COLUMNS_TO_EMBED:
    data = data.withColumn(column, F.when(F.col(column).isNull(), "N/A").otherwise(F.col(column)))

for column in RAW_DATA_STRING_ARRAY_FIELDS:
    data = data.withColumn(
        column, F.regexp_replace(F.regexp_replace(data[column], r"[\[\]'\s]", ""), r",", " ")
    )

for column in RAW_DATA_COLUMNS_TO_DROP:
    data = data.drop(F.col(column))
```

Implementation (cont.)

- **Step 2**: Train Word2Vec Model
  - Word2Vec generates dense embeddings for words by analyzing their contextual relationships.
  - Combine title and description embeddings for better accuracy.

```
tokenizers = [
    Tokenizer(inputCol=column, outputCol=f"{column}{TOKENS_COLUMN_SUFFIX}")
    for column in COLUMNS_TO_EMBED
]

word2Vecs = [
    Word2Vec(
        vectorSize=100,
        minCount=0,
        inputCol=f"{column}{TOKENS_COLUMN_SUFFIX}",
        outputCol=f"{column}{EMBEDDING_COLUMN_SUFFIX}",
    )
    for column in COLUMNS_TO_EMBED
]

stages = tokenizers + word2Vecs
word_embedding_pipeline = Pipeline(stages=stages)

model = word_embedding_pipeline.fit(data)
result = model.transform(data)
```

Implementation (cont.)

▪ Step 3: Calculate Cosine Similarity

Define the Prediction Function:

- For each book with missing categories, compare its **embeddings** (title/description) to reference embeddings.
- Combine cosine similarity scores from title and description embeddings using **weights**.

```
similarities = torch.nn.functional.cosine_similarity(
    target_embedding,
    (
        title_reference_embeddings
        if column == "Title_embeddings"
        else description_reference_embeddings
    ),
    dim=1,
)

if combined_similarities is None:
    combined_similarities = similarities * weight
else:
    combined_similarities += similarities * weight
```

Implementation (cont.)

▪ Step 4: Assign Categories

Retrieve Top-10 Similar Books and Determine the Most Common Category:

- Use `torch.topk()` to get the indices of the **top-10 most similar books**.
- Extract the categories of these top books.
- Use `Counter` to identify the **most frequent category** among the top-10 similar books.

```
top_k_indices = torch.topk(combined_similarities, k=10).indices.numpy()
top_categories = []
for i in top_k_indices:
    top_categories.extend(reference_categories[i].split(" "))

most_common_category = Counter(top_categories).most_common(1)[0][0]
```

Implementation (cont.)

▪ **Step 4**: Assign Categories (Cont.)

Wrap into a PySpark UDF:

- The prediction function is wrapped into a **User-Defined Function (UDF)** for distributed computation across books.

```
predict_category_udf = F.udf(
    lambda x: predict_category(
        x,
        title_reference_embeddings,
        description_reference_embeddings,
        reference_categories,
        COLUMNS_TO_USE_FOR_CATEGORY_FILLING,
    ),
    StringType(),
)
```

Implementation (cont.)

▪ Step 4: Assign Categories (Cont.)

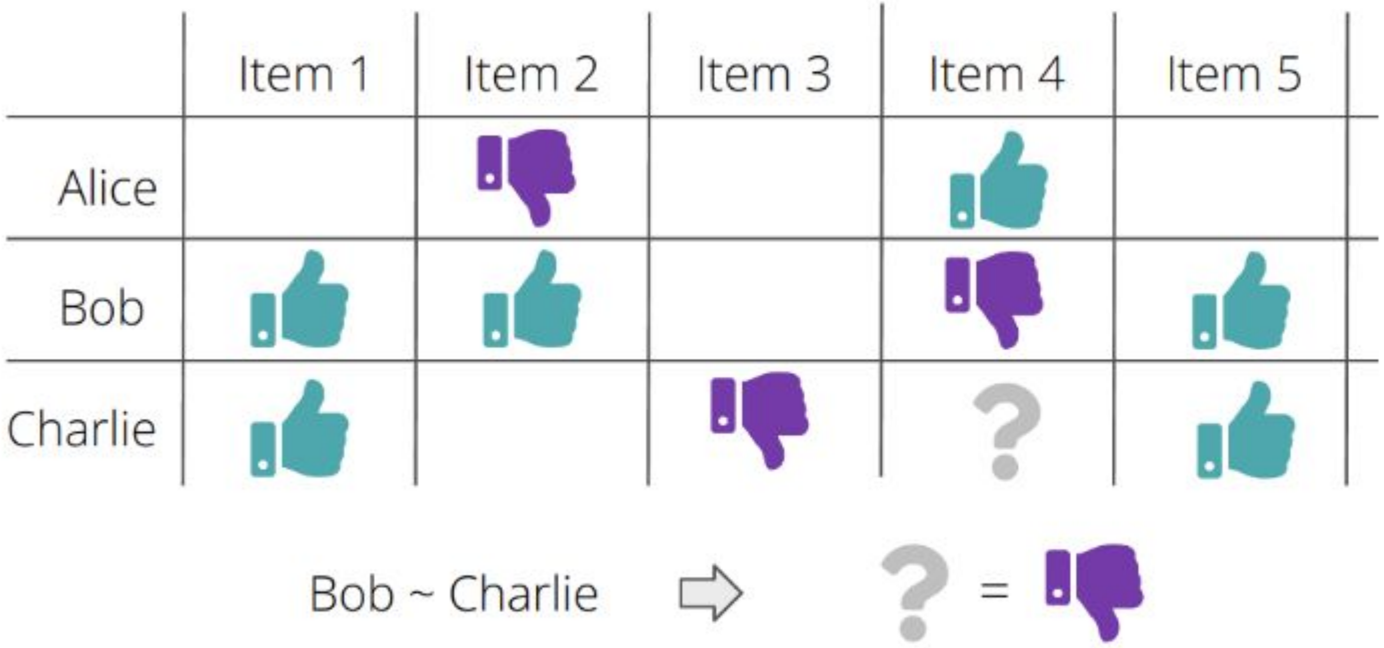
Update the DataFrame with Predicted Categories:

- Join the predicted categories back to the original DataFrame.
- Replace missing categories with predicted values.

```
updated_df = df.join(
    filled_empty_df.select("Title", F.col("predicted_category").alias("predicted_category")),
    on="Title",
    how="left_outer"
).withColumn(
    "categories",
    F.when(F.col("predicted_category").isNotNull(), F.col("predicted_category"))
    .otherwise(F.col("categories"))
).drop("predicted_category")
```

Collaborative Filtering

Introduction



Bob and Charlie have a similar taste. Therefore they are likely to agree on item 4 as well.

Data Pre-processing

- **Filtering Users with Insufficient Records:** Retained only users with 10 or more records(reviews).
  - After filtering, we have a dataset with **819551** reviews
- **User and Book Indexing:** Encoded raw User IDs and Book IDs into numerical indices to align with the input requirements of the ALS model.

categories	UserId	BookId	rating
Knowledge Theoryof	3	18089	5.0
Knowledge Theoryof	255	18089	5.0
Knowledge Theoryof	241	18089	4.0
Knowledge Theoryof	7621	18089	5.0
Knowledge Theoryof	7513	18089	2.0
Knowledge Theoryof	26179	18089	1.0
History	774	90749	5.0
Fiction	329	8332	4.0
Fiction	4866	8332	4.0
Fiction	10	8332	4.0

- **Train-Test Split:** The dataset was split into training (80%) and testing (20%) sets to evaluate the model's generalization capability.
  - Training Dataset Size: 646021
  - Test Dataset Size: 161504

ALS-Model

Train

Hyperparameters

- **Rank**: Set to 100 to capture a higher-dimensional latent representation of the sparse dataset.
- **Regularization Parameter (regParam)**: Set to 0.1 to balance model complexity and prevent overfitting.
- **Max Iterations**: Set to 10 to ensure convergence.
- **Cold Start Strategy**: Configured to drop NaN predictions to avoid issues when recommending books to users not seen during training.

Evaluation

Root Mean Squared Error between ratings and predicted ratings: 0.8175

Auxiliary: Category Label Hybrid Filtering

- **Why?**
  - Aid collaborative filtering (ALS) with content-based filtering, effectively generating the recommendations to match users preference
  - Address the limitations of purely behavior-based recommendations
- **How it works?**
  - Identifies each user's "liked book categories" by calculating the average of ratings for each book category and then filter based on a predefined threshold
  - Filter the ALS-generated recommendations so we don't recommend book in the categories that the user rated poorly before

Category Label Hybrid Filtering

Top Category:

UserId	top_categories
12	[Art, Arcticregio...
22	[Mathematics, Tra...
26	[Cooking, Governe...
27	[Audioequipmentin...
28	[British, William...
31	[LiteraryCriticis...
34	[PerformingArts, ...
44	[History, Biograp...
47	[Pets, Oceania, C...
53	[Religion, POETRY...

only showing top 10 rows

ALS Prediction:

UserId	recommendations
26	[{99733, 6.88063}, {108228, 7.3169246},
27	[{99733, 6.2930837}, {108228, 6.833697},
28	[{99733, 5.44227}, {108228, 6.832974}, {
31	[{108228, 5.775279}, {107438, 5.166066},
34	[{99733, 5.9463935}, {108228, 6.7815876}
53	[{99733, 6.7036986}, {108228, 7.3670263}
65	[{99733, 6.4295197}, {108228, 6.959325},
76	[{99733, 5.4956126}, {108228, 6.2136197}
78	[{99733, 6.435916}, {108228, 7.21666}, {
81	[{108228, 6.0128803}, {13353, 5.2280684}
85	[{99733, 6.683186}, {108228, 6.9425855},

Final Recommendation:

User_id	Title
A1T17LMQABMBN5	Elephant hunting ...
A1T17LMQABMBN5	Penny Plain [Hard...
A1T17LMQABMBN5	Testament: The Bi...
A1T17LMQABMBN5	Thompson Chain-Re...
A1T17LMQABMBN5	Teacup Full of Roses
A1T17LMQABMBN5	Twin Stories: The...
A1T17LMQABMBN5	Sum & Substance: ...
A1T17LMQABMBN5	Mary Had a Little...
A1T17LMQABMBN5	Hey Diddle Diddle
A1T17LMQABMBN5	The Fourth Networ...
A1T17LMQABMBN5	The Fourth Networ...
A1T17LMQABMBN5	The Fourth Networ...
A1T17LMQABMBN5	The Fourth Networ...
AHXAPVSHPJ6OJ	Elephant hunting ...
AHXAPVSHPJ6OJ	Penny Plain [Hard...
AHXAPVSHPJ6OJ	Thompson Chain-Re...
AHXAPVSHPJ6OJ	Teacup Full of Roses
AHXAPVSHPJ6OJ	The advance of sc...
AHXAPVSHPJ6OJ	The advance of sc...
AHXAPVSHPJ6OJ	Mary Had a Little...