Book Recommendation System

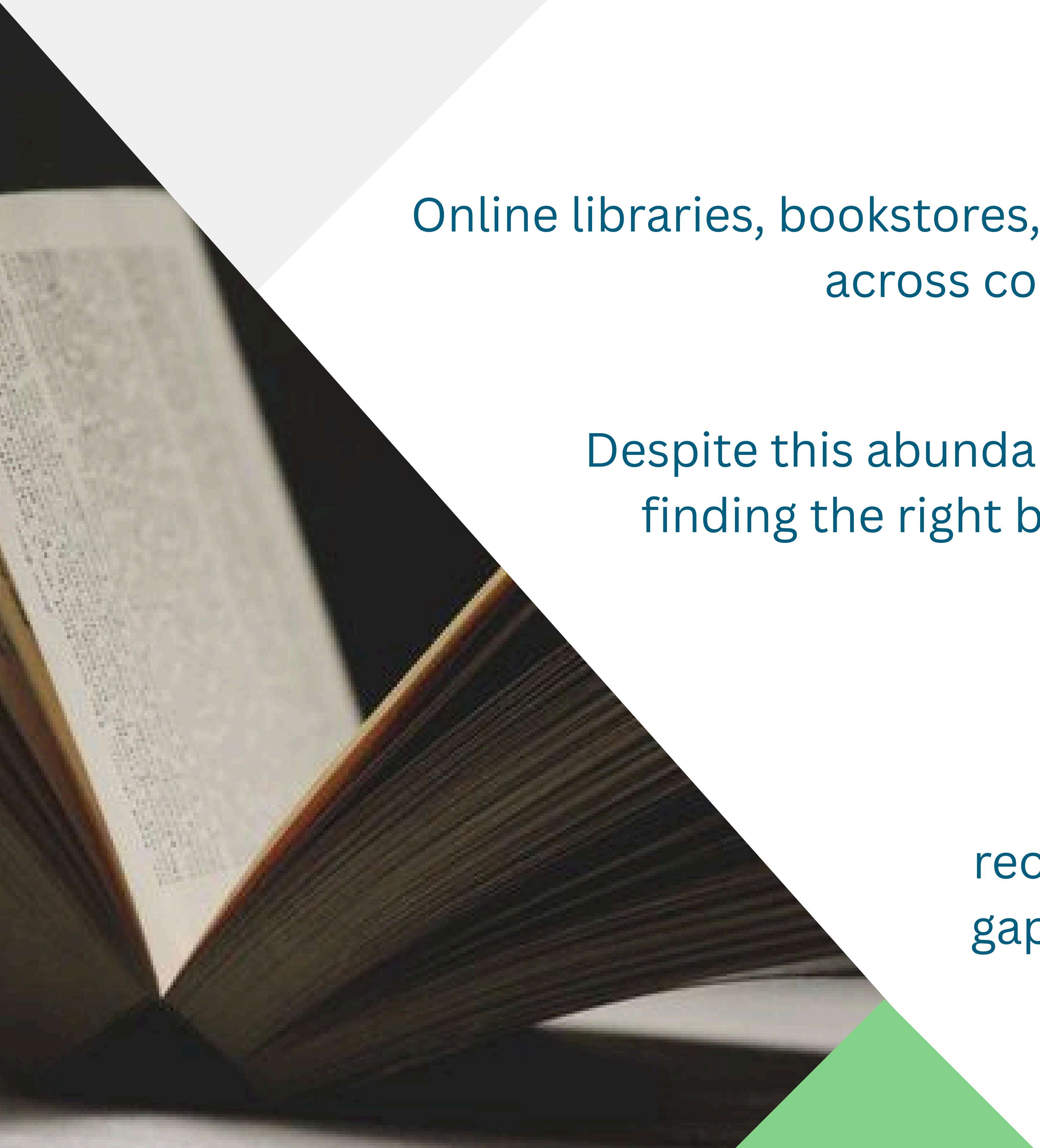
SAMI SIVRI - INSTRUCTOR
JEYHUN FARZULLAYEV - STUDENT

İSTANBUL
TEKNİK ÜNİVERSİTESİ
1773

Plan

- Introduction
- Data Exploration
- Vector Search
- Text Classification
- Sentiment Analysis
- Gradio Interface
- Conclution

Introduction



Online libraries, bookstores, and open-access platforms offer millions of book titles across countless genres, topics, and languages

Despite this abundance, many readers continue to face a common challenge: finding the right book that aligns with their specific interests, emotional preferences, or academic needs.

A solution to this problem lies in intelligent book recommendation systems—tools designed to bridge the gap between user needs and the vast universe of books.

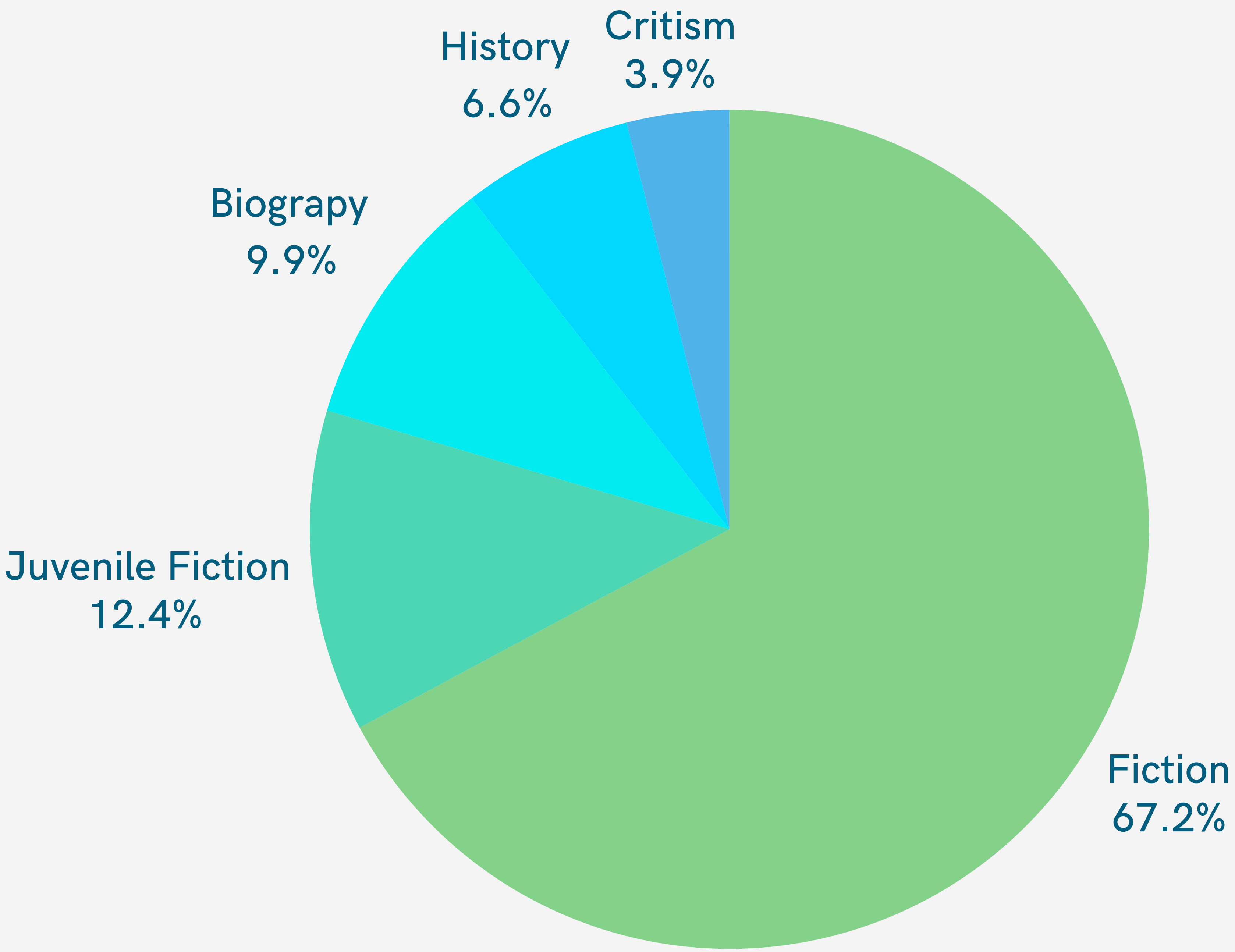
Data Retrival

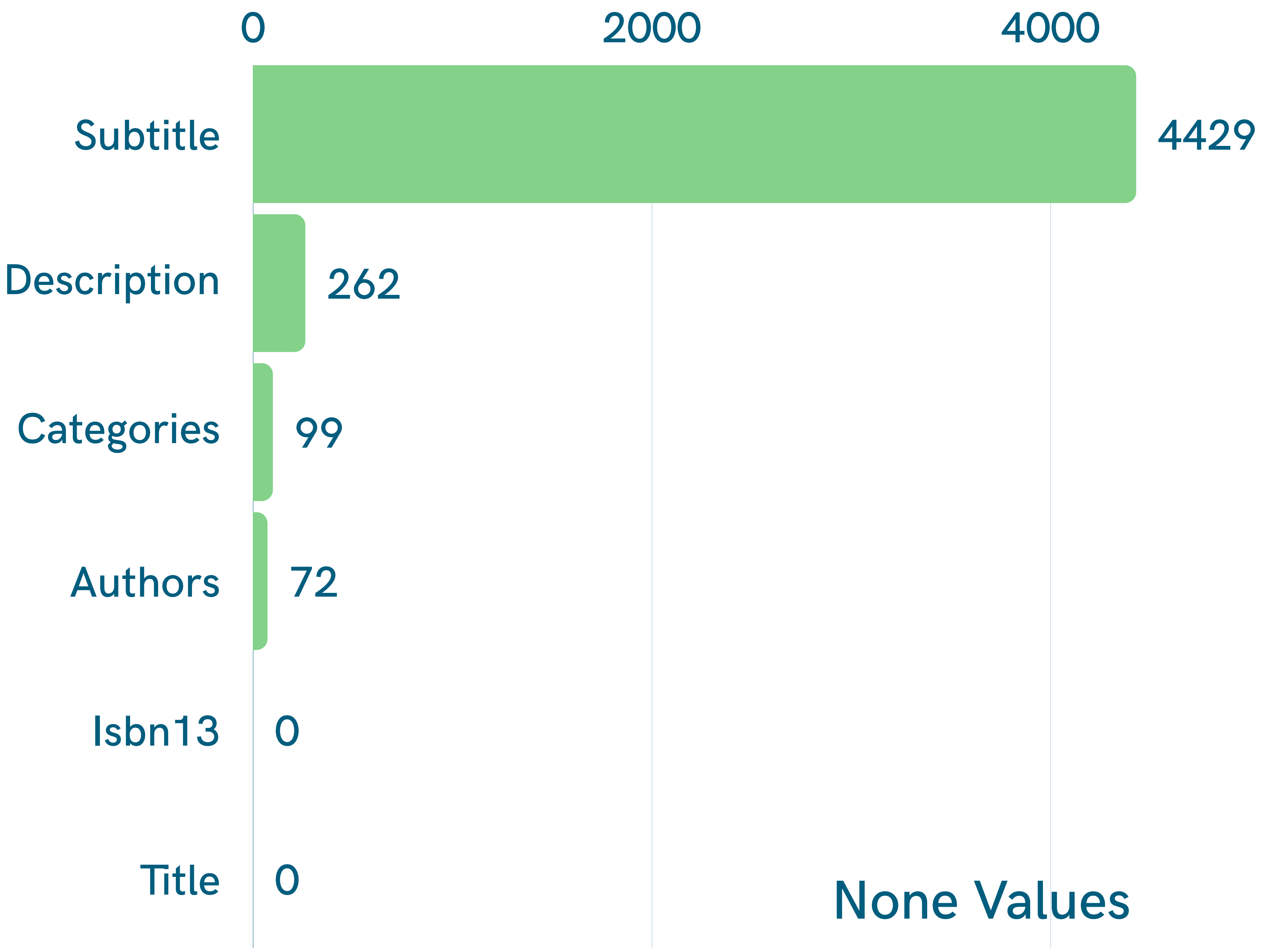
https://www.kaggle.com/datasets/dylanjcastillo/7k-books-with-metadata

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("dylanjcastillo/7k-books-with-metadata")

print("Path to dataset files:", path)
```

- Isbn13
- Isbn10
- title
- subtitle
- author
- description
- category
- thumbnail
- published_year
- avarage_rating
- page_number
- rating_count

Data Analysis

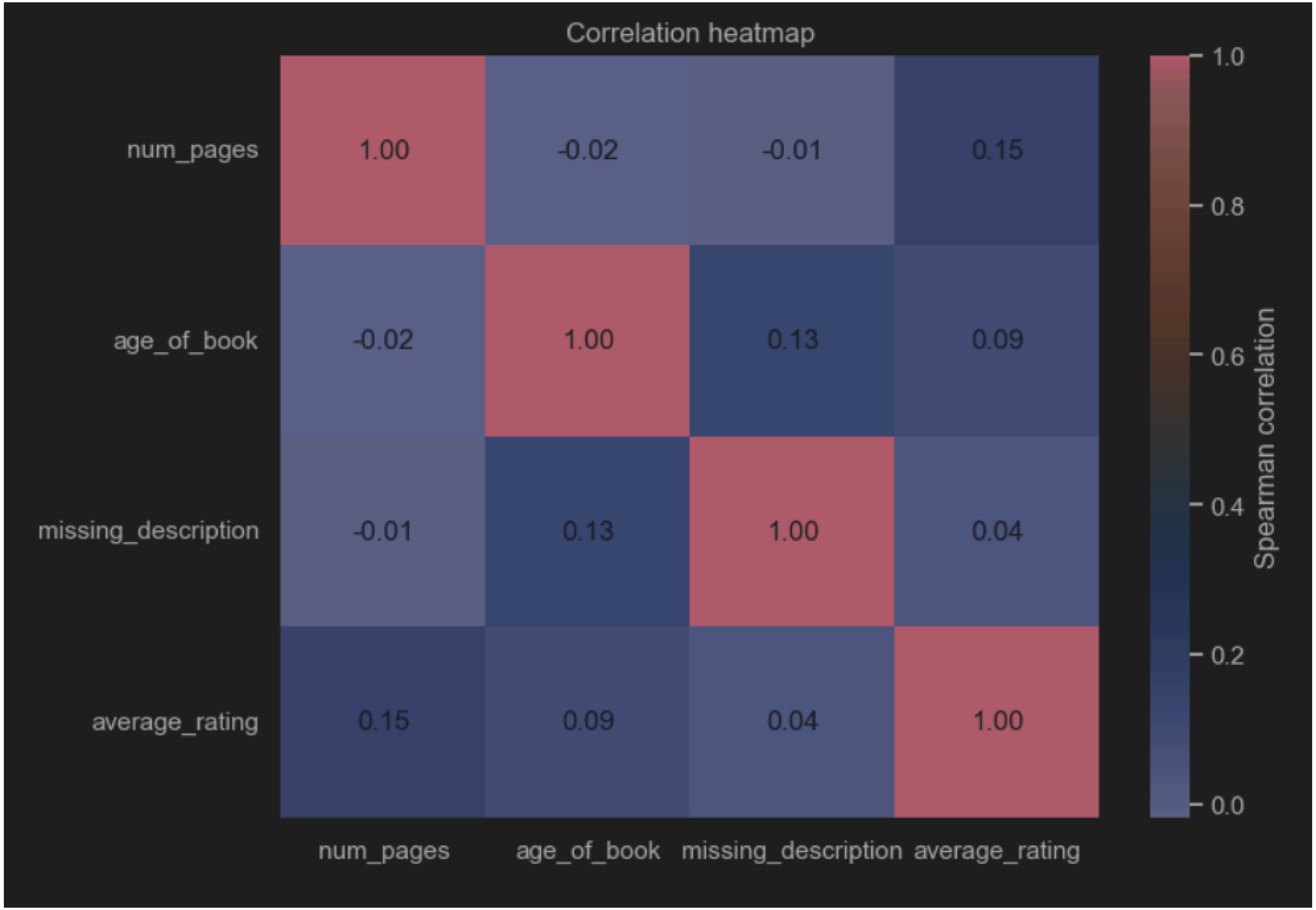




Processing on data

1. Data Analysis Processing
2. Creating age_of_book column
3. Showing correlation matrix
4. Removing Nan values description elements
5. Selection of samples which has length of description feature greater than 25
6. Combining title and subtitle columns
7. Creating tagged_desc column which combines isbn13 and description

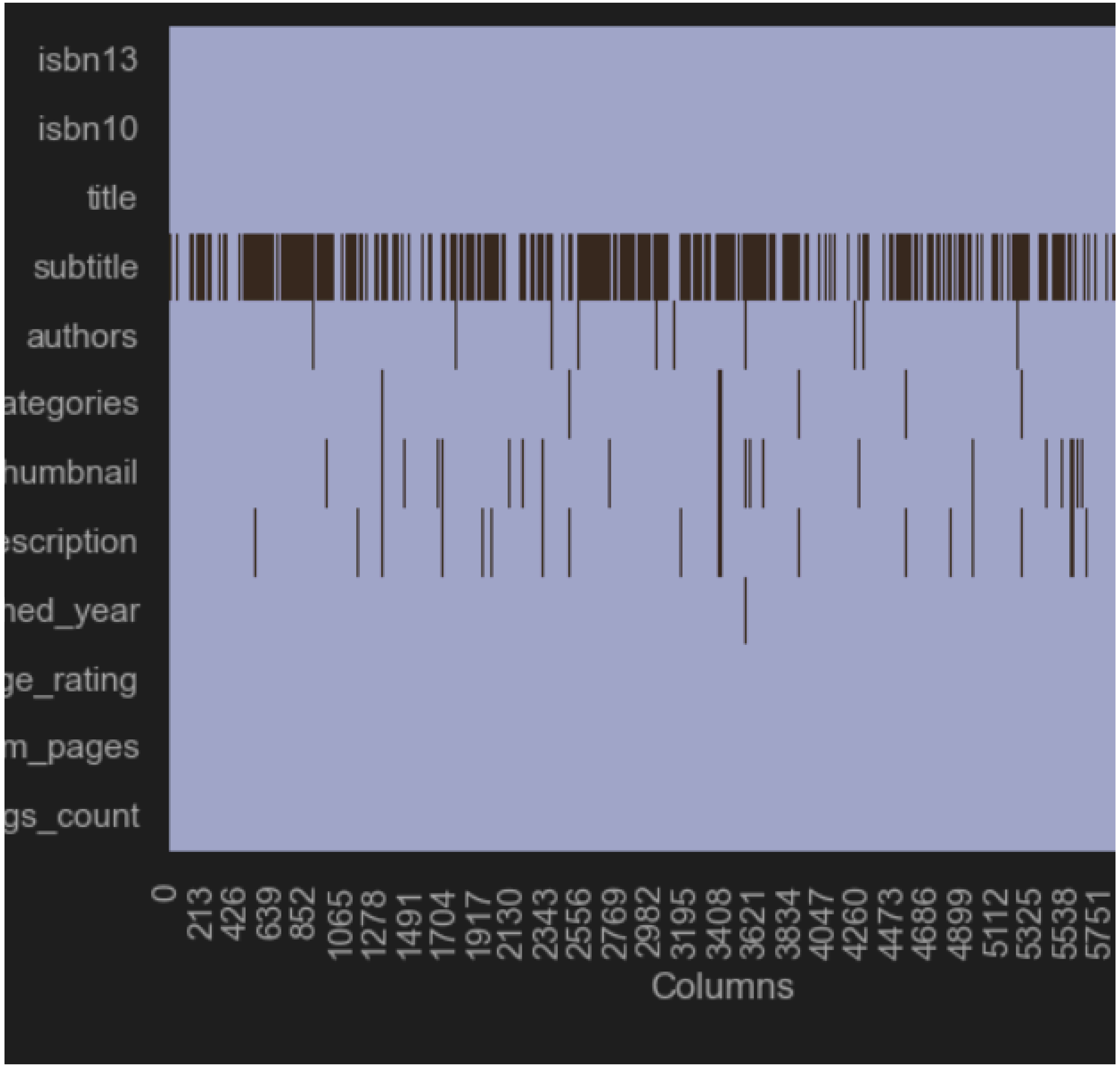
Spearman correlation heatmap



```
columns_of_interest = ["num_pages",
"age_of_book","missing_description", "average_rating"]

correlation_matrix =
books[columns_of_interest].corr(method="spearman")

sns.set_theme(style="white")
plt.figure(figsize=(8,6))
heatmap = sns.heatmap(correlation_matrix,
annot=True,fmt=".2f", cmap="coolwarm", cbar_kws={"label":
"Spearman correlation"})

heatmap.set_title("Correlation heatmap")
plt.show()
```

observing that there is no strong correlation between the variables shown in the correlation matrix.

Data Cleaning



The number of none values of subtiltle is much more.

Combine subtitle and title columns

The number of none values of description is less but exists.

Remove these samples.

The number of samples decreases to 5197

Vector Search

Text Splitter

Create tagged description.txt file

Seperate descriptions into documents(chunks)

HuggingFaceEmbeddings

Embedding with BERT based sentence-transformers/all-MiniLM-L6-v2 model. It create vector for sentences then easy apply for similarity search.

Chroma DB

Creating vector base to keep embedding and it provides quick similarity search.

Similarity Search

In Langchain Chroma uses cosine similarity search. It measures the similarity of two vectors based on the angle between them.

Text Splitter

```
books["tagged_description"].to_csv("tagged_
description.txt",
sep = "\n",
index = False,
header = False)
```

Converting tagged _description column to csv file which contains all descriptions in newline . Index number and header are not allowed.

```
raw_documents = TextLoader("tagged_description.txt",
encoding='utf-8').load()
text_splitter = CharacterTextSplitter(chunk_size=0,
chunk_overlap=0, separator="\n")
documents =
text_splitter.split_documents(raw_documents)
```

Loading tagged description.txt , splitting them into chunks and creating documents from each line;

Hugging Face Embedding

Sentence-Transformer/all-mpnet-base-v2

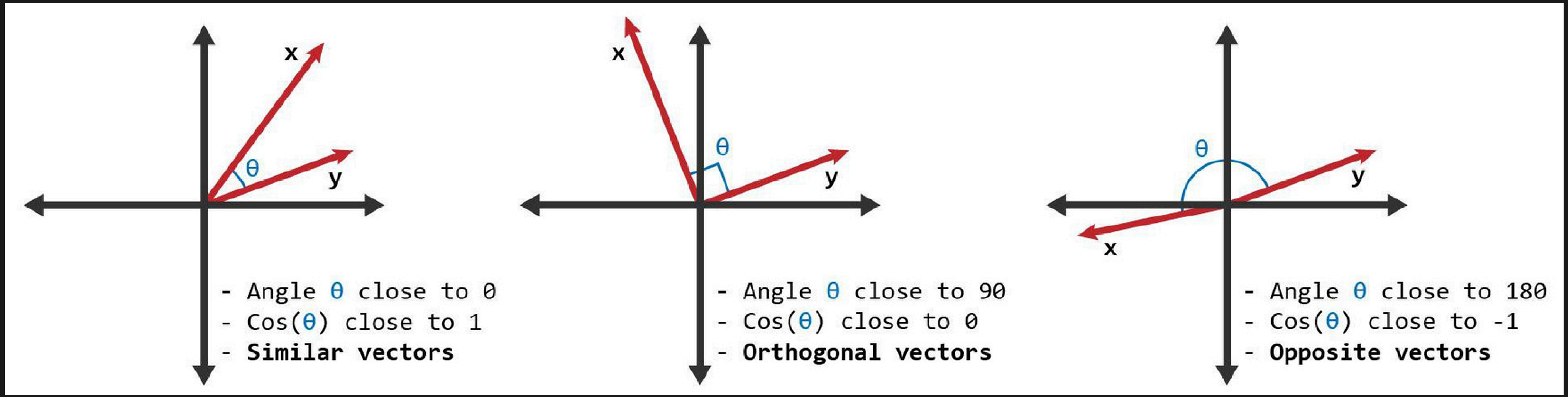
Property	Value
Model Type	Transformer (MPNet-based)
Embedding Dimension	768
Architecture	MPNet (Masked and Permuted Pre-training)
Number of Layers	12
Pre-trained On	Natural Language Inference (NLI) + Semantic Textual Similarity (STS)
Fine-tuned For	Semantic similarity, clustering, sentence embeddings
Languages Supported	English
Performance	One of the **most accurate** sentence transformers (better than MiniLM)
Inference Speed	Slower than MiniLM, but still efficient
Embedding Output	Dense vector representing the sentence meaning (768-dimensional float32)
Hugging Face ID	sentence-transformers/all-mpnet-base-v2
Best For	High-accuracy semantic search, question-answer retrieval, embeddings

Chroma

```
from langchain.vectorstores import Chroma
db_books = Chroma.from_documents(
    documents,
    embedding=embeddings,
    persist_directory="./chroma_books",
)
```

- Stores text embeddings (vectors) along with metadata.
- Supports fast similarity search using cosine or Euclidean distance.
- Helps power semantic search, chatbot memory, retrieval-augmented generation (RAG), and recommendation systems.

Similarity Search



- Angle θ close to 0
- Cos(θ) close to 1
- **Similar vectors**

- Angle θ close to 90
- Cos(θ) close to 0
- **Orthogonal vectors**

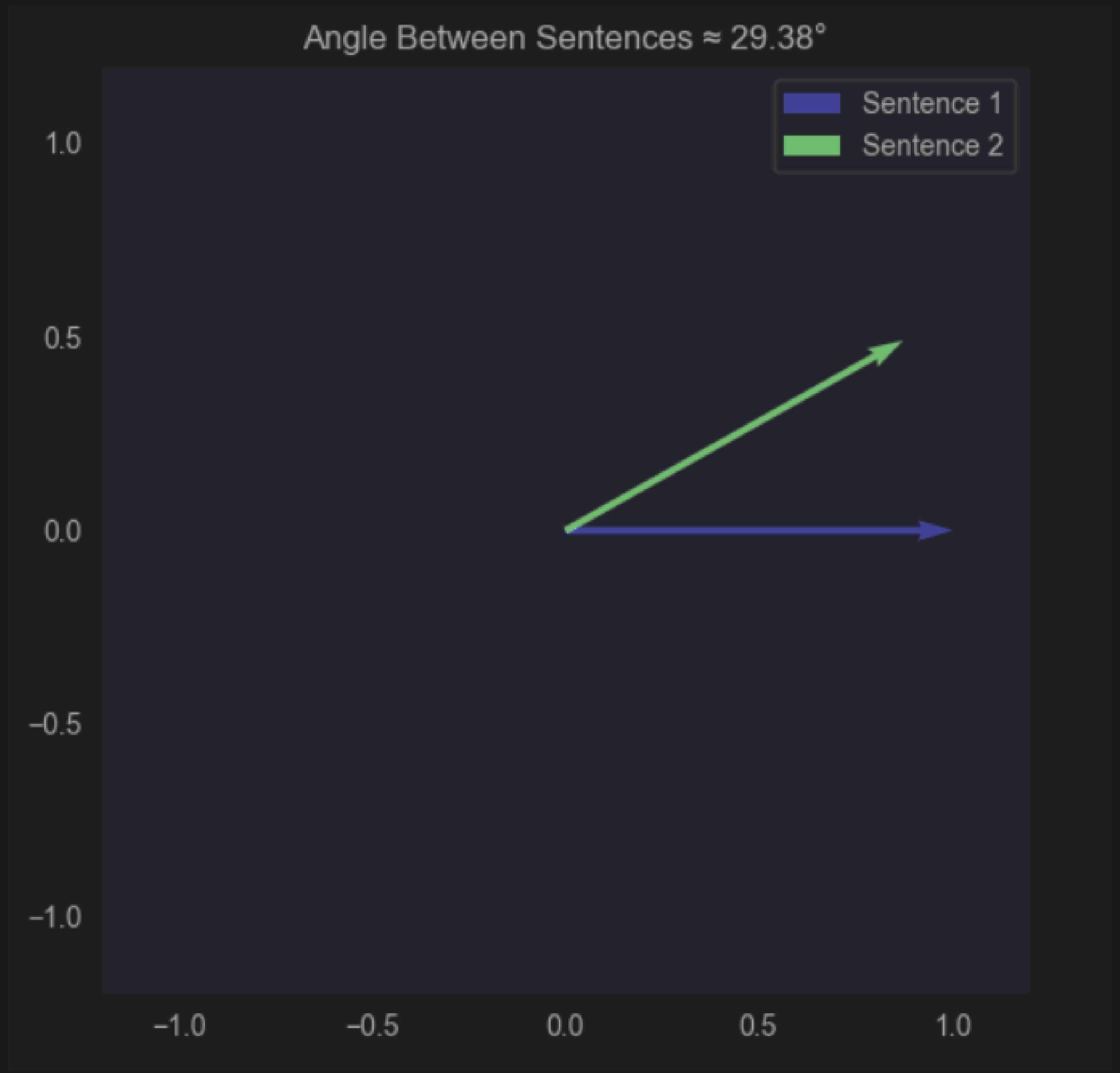
- Angle θ close to 180
- Cos(θ) close to -1
- **Opposite vectors**

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

document 1

"Artificial intelligence is transforming the world."

Embedding 1 (first 10 values):
[0.0891 0.1253 0.0322 0.0924 -0.0451 0.1142 -0.0623
0.0218 0.0157 0.0836]



Angle Between Sentences ≈ 29.38°

document 2

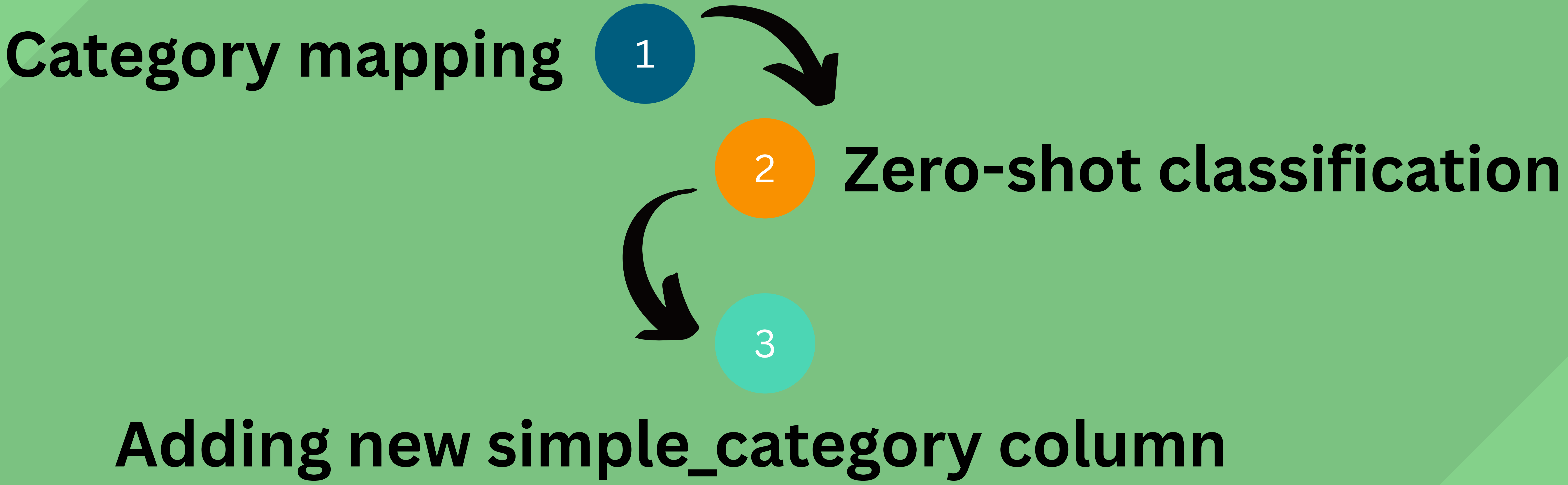
"AI is changing the world."

Embedding 2 (first 10 values):
[0.0879 0.1267 0.0301 0.0903 -0.0446 0.1135
-0.0602 0.0224 0.0163 0.0842]

Cosine Similarity: 0.8708

ChromaDb uses cosine similarity to find distance between vectors.

Text Classification



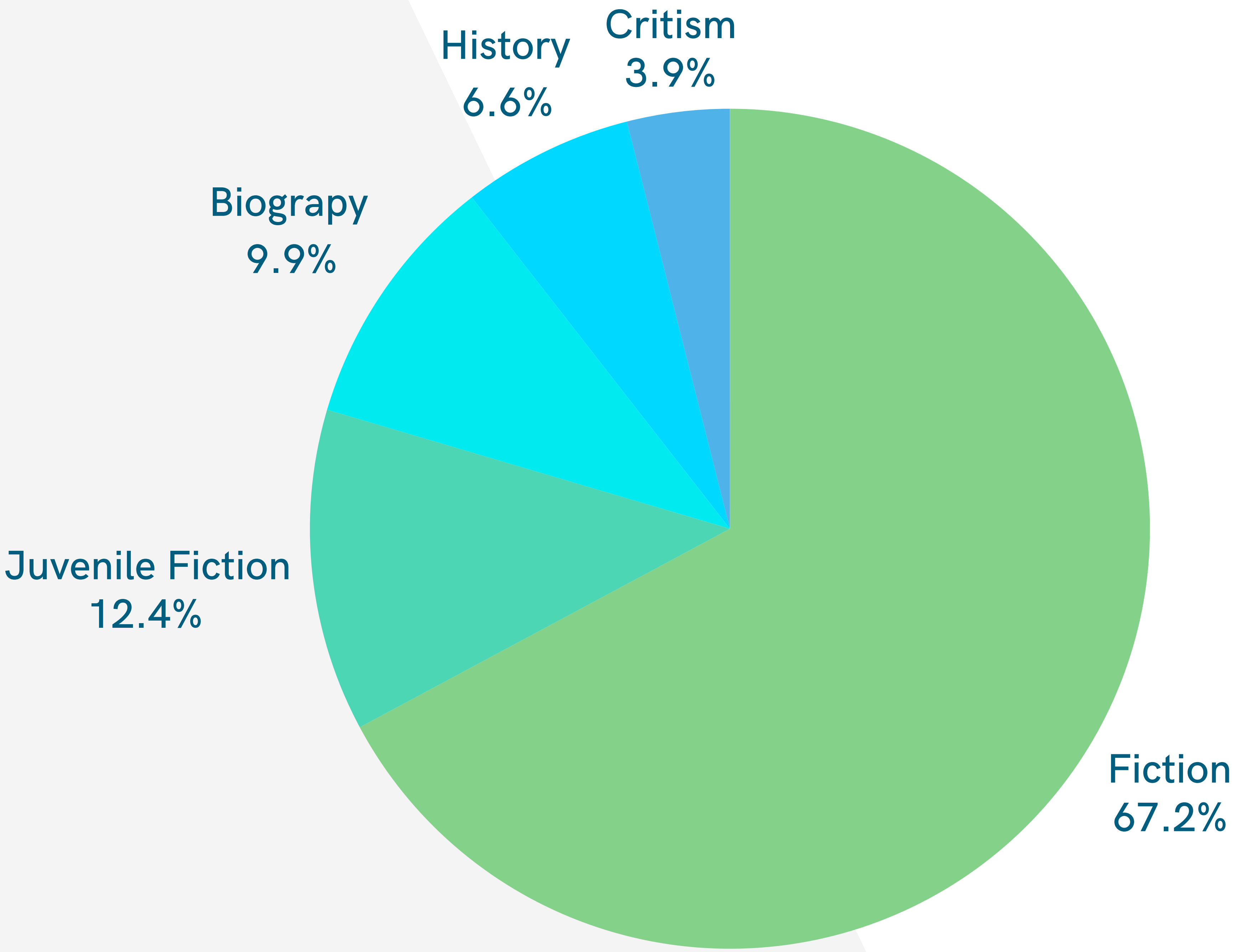
1 Category mapping

2 Zero-shot classification

3 Adding new simple_category column

Category Mapping

```
category_mapping = {'Fiction' : "Fiction",
'Juvenile Fiction': "Children's Fiction",
'Biography & Autobiography': "Nonfiction",
'History': "Nonfiction",
'Literary Criticism': "Nonfiction",
'Philosophy': "Nonfiction",
'Religion': "Nonfiction",
'Comics & Graphic Novels': "Fiction",
'Drama': "Fiction",
'Juvenile Nonfiction': "Children's Nonfiction",
'Science': "Nonfiction",
'Poetry': "Fiction"}

books["simple_categories"] = books["categories"].map(category_mapping)
```



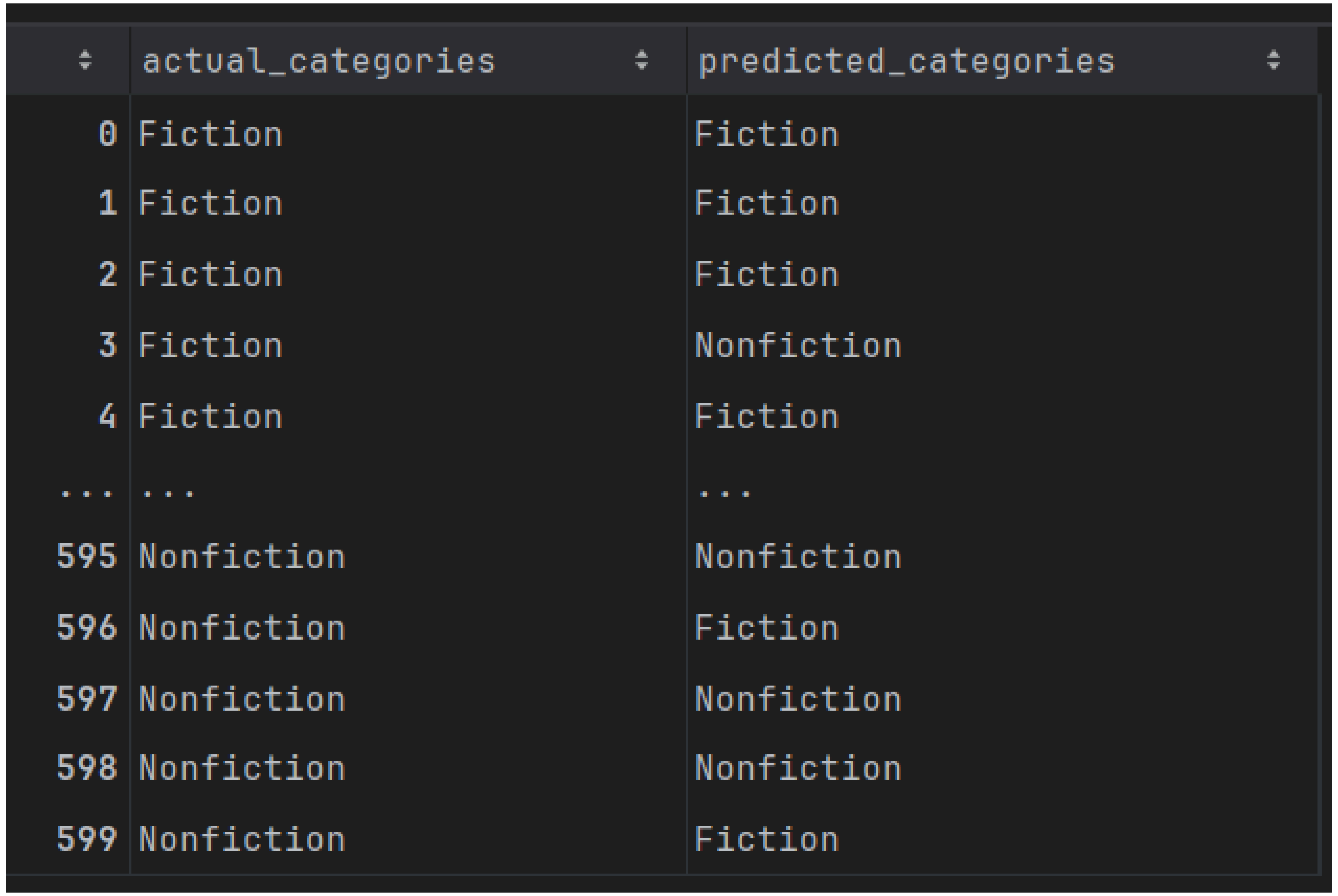
Zero-shot Classification

facebook / bart-large-mnli

```
pipe = pipeline("zero-shot-classification",
          model="facebook/bart-large-mnli",
                    device=0)
```

$$\frac{\text{predictions\_df["correct\_prediction"].sum()}}{\text{len(predictions\_df)}}$$

Evaluation of Classification : 0.78



```
     actual_categories        predicted_categories
0    Fiction                  Fiction
1    Fiction                  Fiction
2    Fiction                  Fiction
3    Fiction                  Nonfiction
4    Fiction                  Fiction
...  ...                      ...
595  Nonfiction               Nonfiction
596  Nonfiction               Fiction
597  Nonfiction               Nonfiction
598  Nonfiction               Nonfiction
599  Nonfiction               Fiction
```

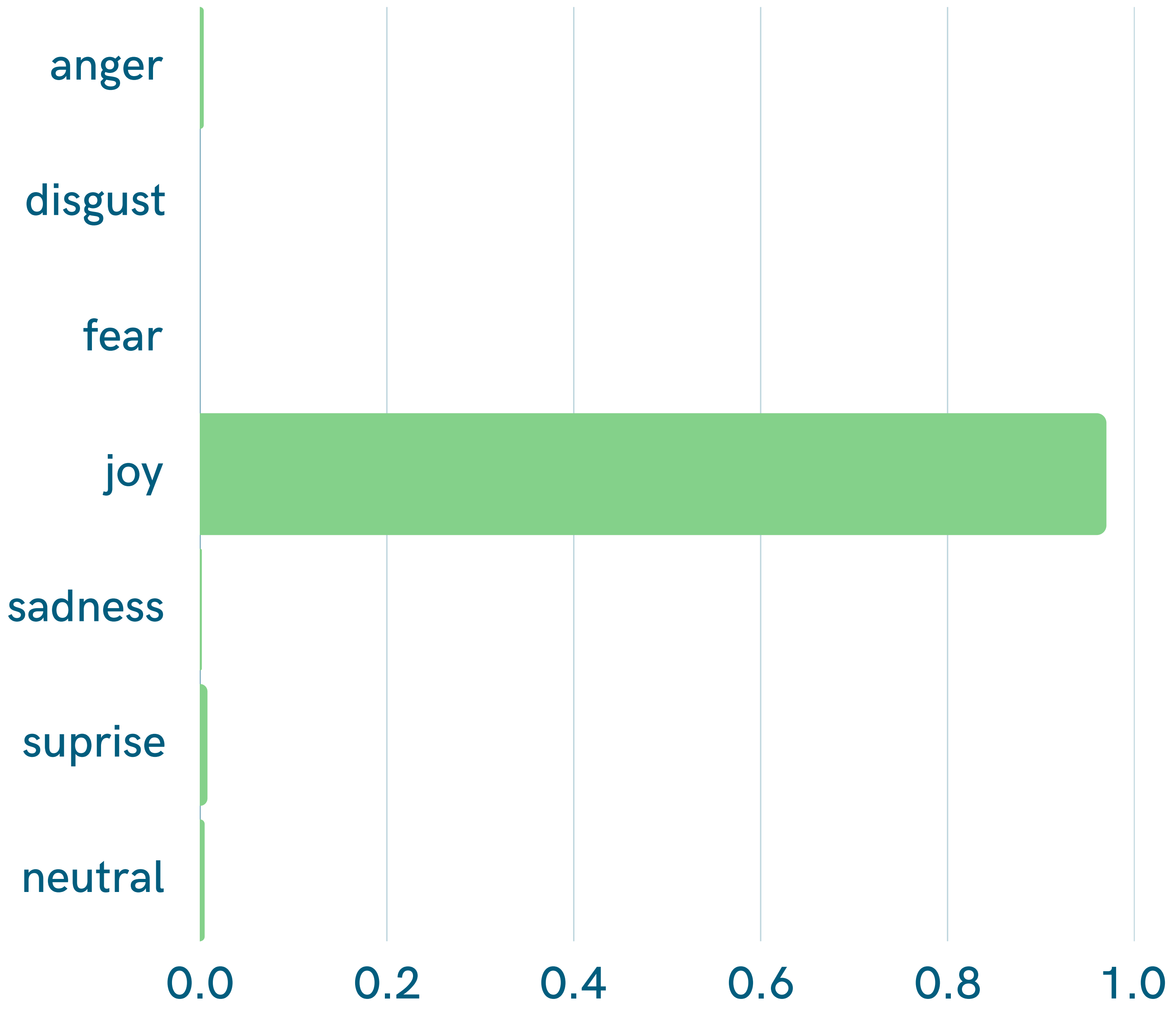
Sentimen Analysis

"j-hartmann/emotion-english-distilroberta-base"

```
classifier = pipeline("text-classification",
          model="j-hartmann/emotion-english-
                distilroberta-base",
                      top_k = None,
                       device = 0)
```

"anger", "disgust", "fear", "joy", "sadness", "surprise", "neutral"

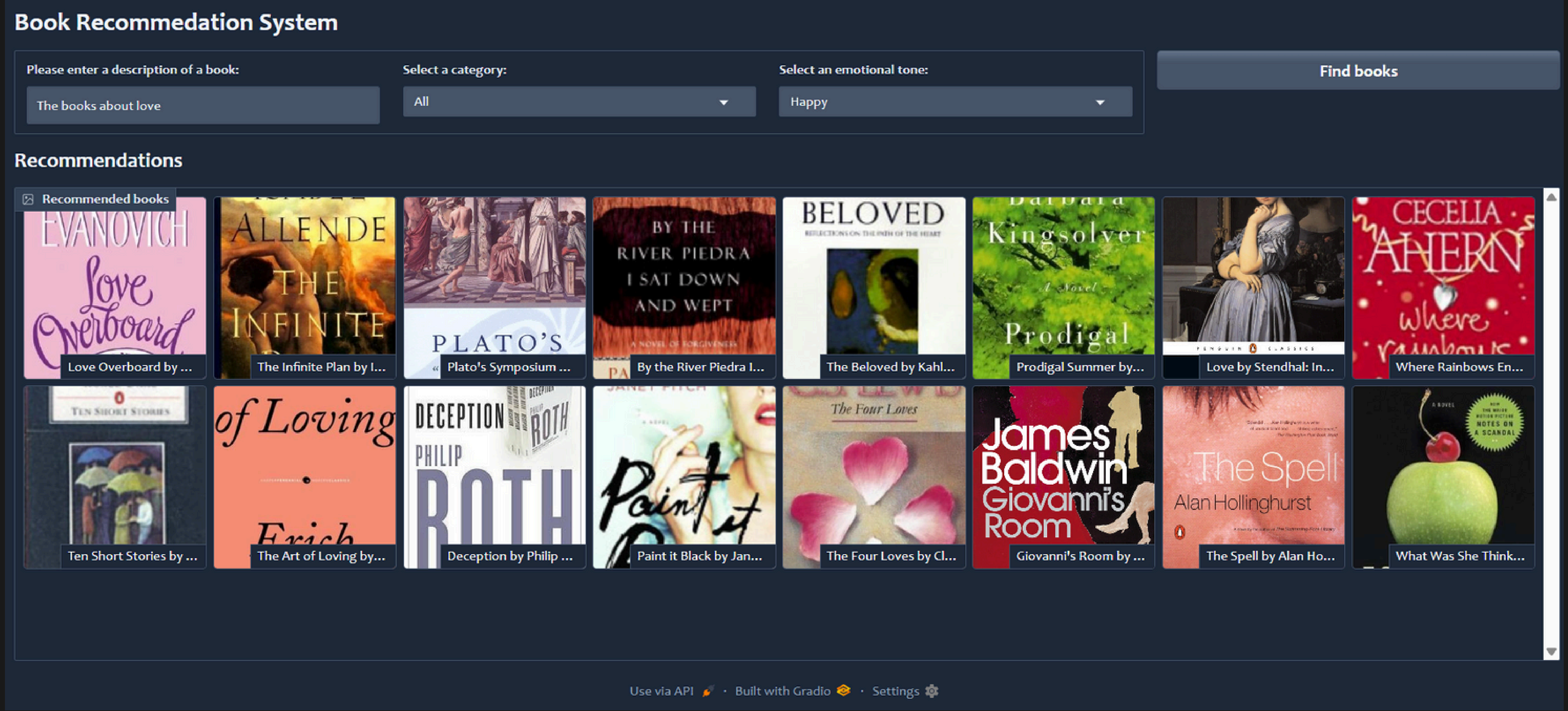
classifier("I love this!")



gradio

Gradio is an open-source Python library that allows you to easily create web-based user interfaces for your machine learning models, functions, or data science workflows — all with just a few lines of code.

📱 Build interactive demos (input/output interfaces for ML models)
🌐 Share your model online via a public URL instantly
🧪 Test your model with different inputs (text, images, audio, etc.)
🛠️ Deploy prototypes without needing front-end skills

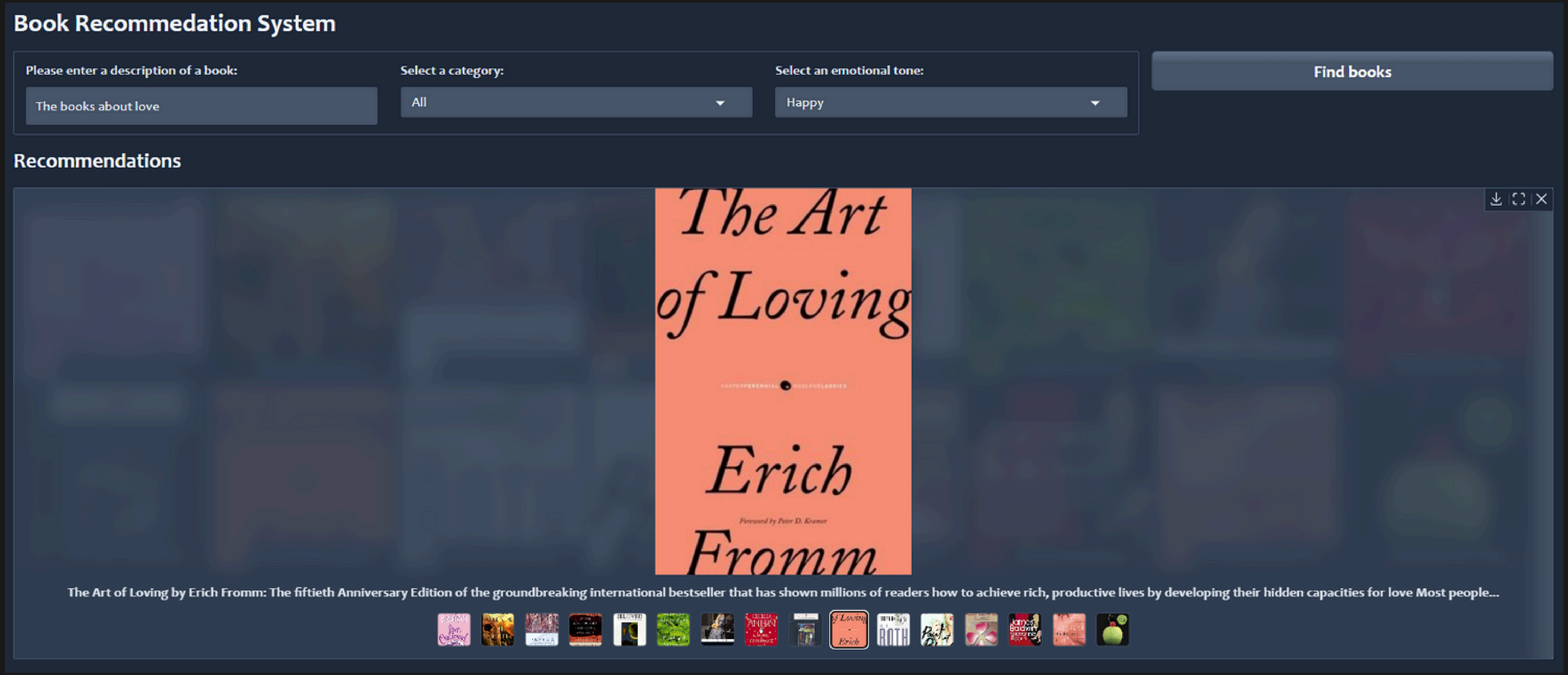


Book Recommedation System

Please enter a description of a book: The books about love

Select a category: All

Select an emotional tone: Happy



Book Recommedation System

Please enter a description of a book: The books about love

Select a category: All

Select an emotional tone: Happy

The Art of Loving by Erich Fromm: The fiftieth Anniversary Edition of the groundbreaking international bestseller that has shown millions of readers how to achieve rich, productive lives by developing their hidden capacities for love Most people...

Limitations

Limited Dataset Size

Dataset contains approximately 7,000 books.

Quality of Results

Recommended books may not fully match the user's intent due to this limitation.

Embedding Methods

More advanced embedding models like OpenAI's GPT Embeddings or DeepSeek Embeddings could potentially provide more accurate semantic understanding of user queries and book descriptions.

Unbalanced Categories

The distribution of book categories in the dataset is uneven.

Conclution

This project showcases an intelligent book recommendation system that integrates sentiment analysis, semantic search, and NLP to deliver personalized, emotion-aware suggestions. By leveraging tools like HuggingFace Transformers, Sentence Transformers, and Gradio, we built a smooth pipeline from raw data to a user-friendly interface. The system aligns recommendations with user emotions and query meaning, enhancing relevance and user experience. It lays a strong foundation for future improvements, such as incorporating feedback, reading history, or multimodal inputs.

Hugging Face

PC

Thank You

kaggle

gradio