Amazon Product Trend Analyzer: Project Documentation

A. Project Overview

Goal: The goal of this project is to uncover market trends and product opportunities by analyzing co-purchasing relationships and sales performance within Amazon's ecosystem of products.

Dataset:

- **Name:** Amazon Product Co-Purchasing Network (2003)
- **Source:** Stanford SNAP Datasets
- **Size:** 450MB (compressed), 1.8GB (uncompressed)
- **Scope:** 548,552 products and 1,788,725 co-purchase relationships

This dataset captures the structure of consumer purchasing behavior through “similar products” relationships and sales rank metadata.

B. Data Processing Pipeline

Loading Process:

- Parsed dataset **line-by-line** from raw text to extract essential attributes like:
    - ASIN (unique product ID)
    - Title (product name)
    - Group (product category)
    - Salesrank (relative popularity)

- I developed an in-memory graph structure to represent the similarity relationships between products.

Data Transformations:

- **Salesrank**: The data was cleaned and cast to an integer in order to make it consistent and easy to analyze.

- **Graph Edges**: Bidirectional connections were built between "similar products" to allow unrestricted access to those products.

- **Filtering**: A number of nodes were removed due to invalid or missing sales rank values in order to maintain the integrity of the data.

Through the use of structured processing methods, it was possible to query and visualize product trends efficiently.

C. Code Structure

The codebase is organized into three main modules: *loader.rs*, *analyzer.rs*, and *main.rs*. This structure separates data loading, analysis, and execution logic, making the project easier to read, test, and extend.

1. Modules
   a) Loader.rs - Parses Amazon product data from file into a graph structure with product metadata.
   b) Analyzer.rs - Analyzes product relationships, trends and market opportunities through algorithms.
   c) Main.rs - Handles data loading, analysis execution and result presentation
   d) Analyzer_test.rs - Unit tests for analyzer functions
   e) Loader_test.rs - Unit tests for data loading and parsing functionality.

2. Key Functions & Types (Structs, Enums, Traits, etc)
   a) Product Struct - Stores core metadata for an Amazon product.
      i) Represents a product in the amazon database. The fields are populated during parsing and serve as the foundation for all analyses.

b) ProductDataset Struct - Container for the loaded dataset and its graph representation.
    i) Nodes represent ASINs, edges represent co-purchasing relationships, and the product's hashmap enables fast ASIN-to-Product lookup.

c) Load_Dataset() Function - Parse raw dataset file into a structured ProductDataset type.
    i) Parses the file line-by-line using a state machine to extract products and relationships, building graph edges for similar products.

d) Detect_trend_clusters() Function - Uses Kosaraju's Strongly Connected Components (SCC) algorithm to identify product clusters.
    i) Automatically finds groups of related products (like book series or matching items) that customers often buy together, focusing only on meaningful groups.

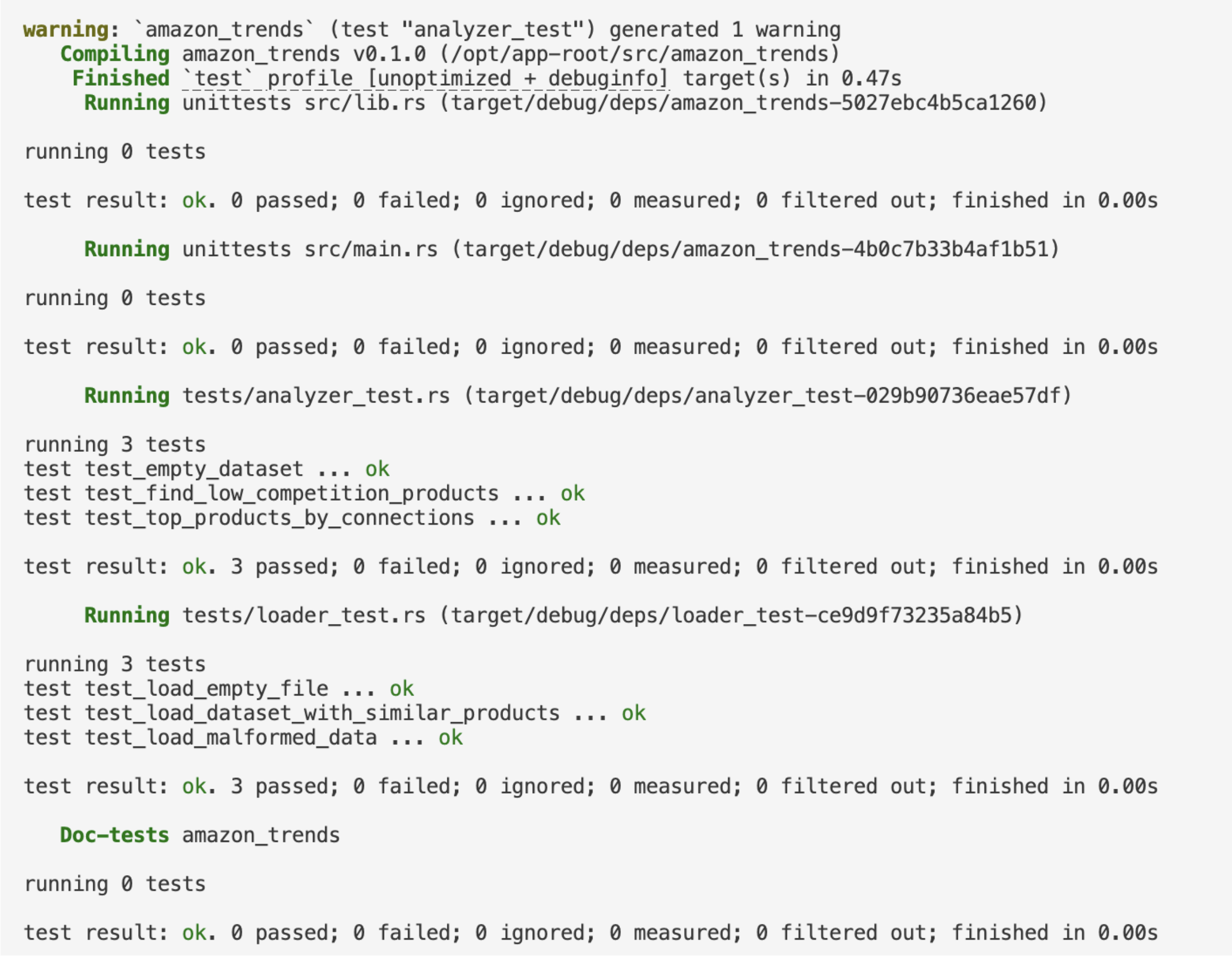
e) Find_low_competition_products() Function - Evaluates products based on their sales rank relative to their cluster size.
    i) Finds products that are popular (good sales rank) but don't have many similar competing products (small cluster size) - these represent the best business opportunities.

f) Analyze_bestsellers() Function - Ranks products by their number of co-purchase connections.
    i) Shows the most frequently co-purchased products. These are the best-selling items that customers often buy together.

3. Main Workflow

1. Load Dataset from amazon-meta.txt
2. Parse Product Relationships into a connection graph
3. Identify Top Products by purchase connections
4. Detect Trend Clusters (groups of frequently co-purchased items)
5. Score Market Opportunities (popular products with low competition)
6. Display Insights:
    a. Best-selling products
    b. Emerging product trends
    c. Recommended opportunities

D. Tests



```
warning: `amazon_trends` (test "analyzer_test") generated 1 warning
   Compiling amazon_trends v0.1.0 (/opt/app-root/src/amazon_trends)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 0.47s
     Running unittests src/lib.rs (target/debug/deps/amazon_trends-5027ebc4b5ca1260)

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running unittests src/main.rs (target/debug/deps/amazon_trends-4b0c7b33b4af1b51)

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running tests/analyzer_test.rs (target/debug/deps/analyzer_test-029b90736eae57df)

running 3 tests
test test_empty_dataset ... ok
test test_find_low_competition_products ... ok
test test_top_products_by_connections ... ok

test result: ok. 3 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running tests/loader_test.rs (target/debug/deps/loader_test-ce9d9f73235a84b5)

running 3 tests
test test_load_empty_file ... ok
test test_load_dataset_with_similar_products ... ok
test test_load_malformed_data ... ok

test result: ok. 3 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

   Doc-tests amazon_trends

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

Why does each test matter:

Analyzer_tests:

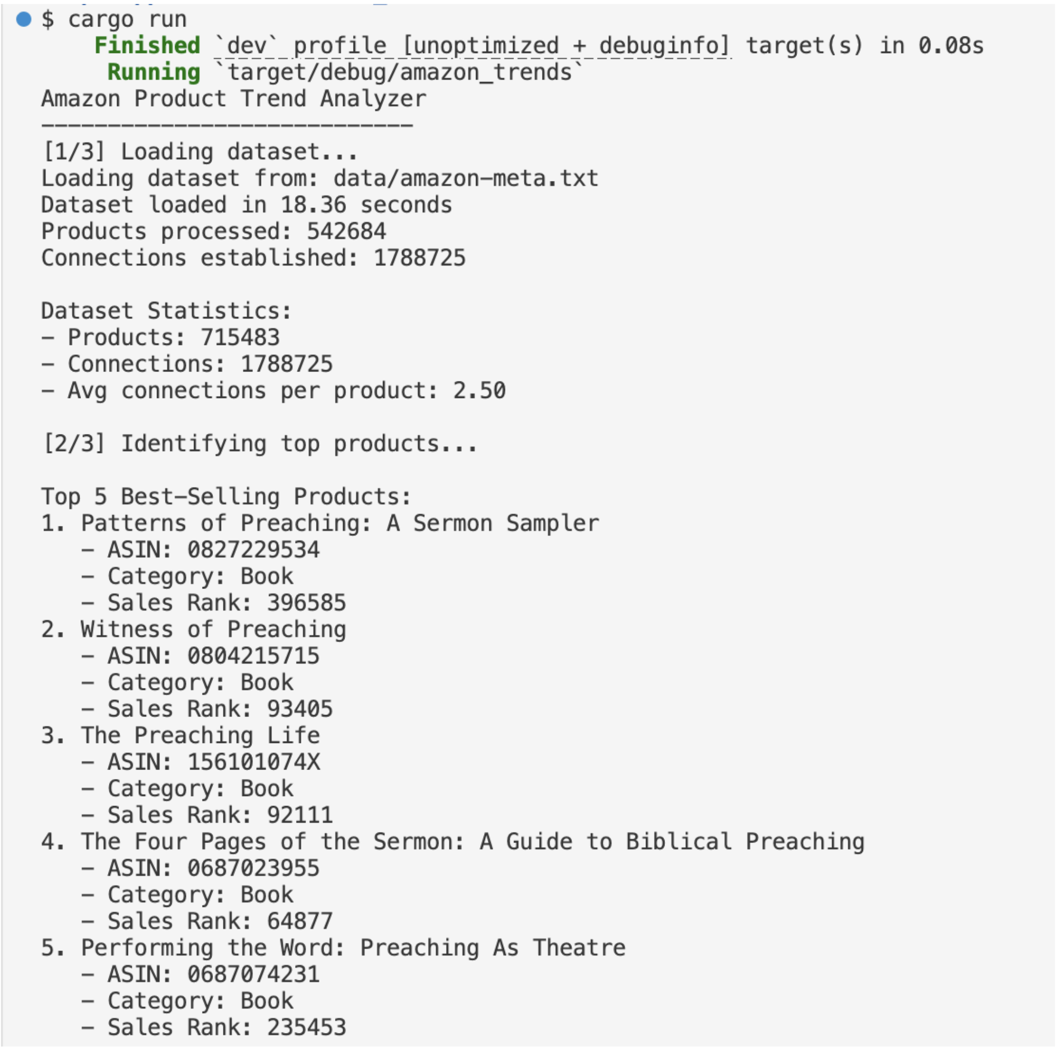
- Test_top_products_by_connections → Ensures that the best-selling products are correctly identified based on the purchase connections we make.
- Test_find_low_competition_products → Verifies the accuracy of our scoring system and identifies high-potential new products with little competition
- Test_empty_dataset → This ensures that empty inputs will be handled gracefully, preventing crashes in production environments when this is necessary.

Loader_tests:

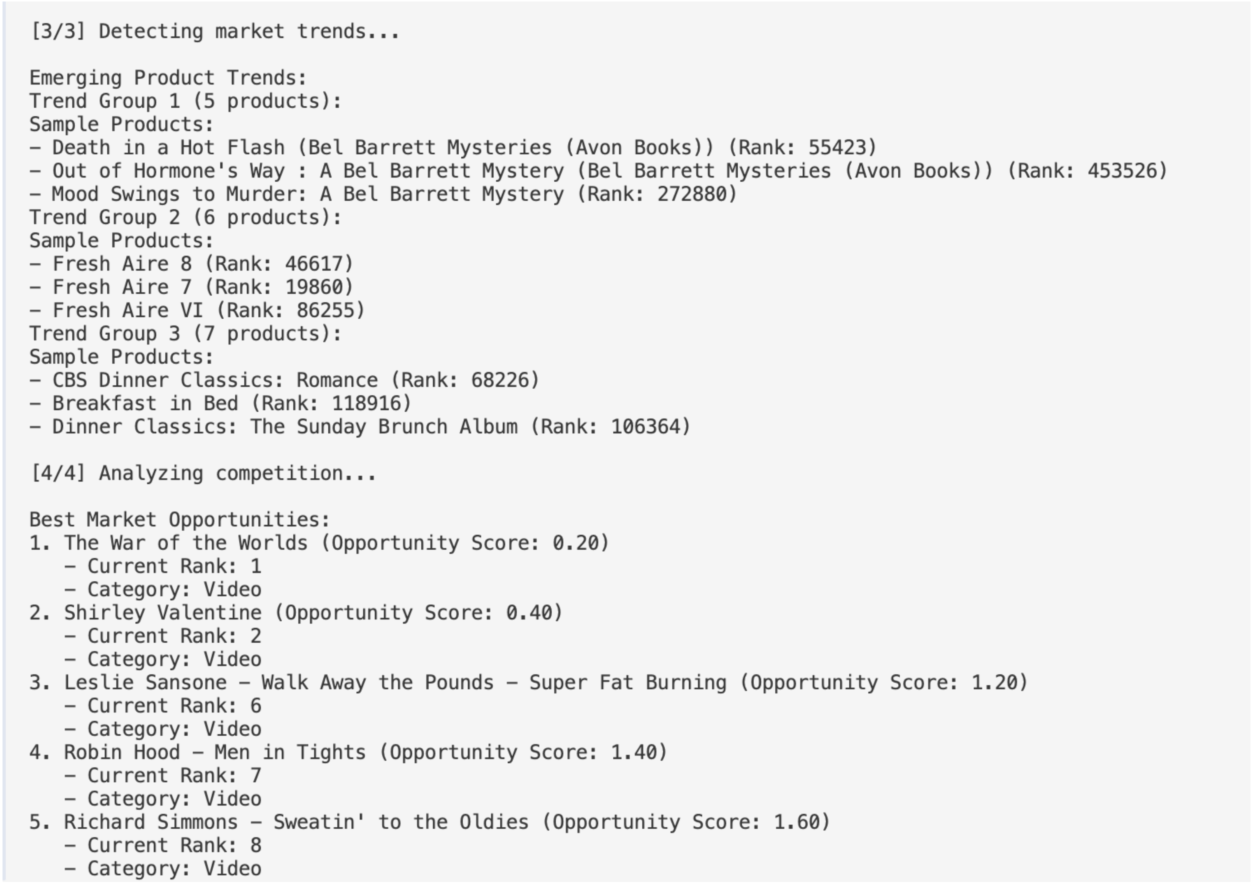
- Test_load_dataset_with_similar_products → Demonstrates that we have captured the relationships between products that are frequently purchased together.
- Test_load_malformed_data → Confirms that the parser rejects invalid formats (missing ASIN/title).
- Test_load_empty_file → Provides graceful error handling for input files that are empty.

E. Results and interpretations

Complete program output:



```
● $ cargo run
      Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.08s
       Running `target/debug/amazon_trends`
  Amazon Product Trend Analyzer
  ----------------------------
  [1/3] Loading dataset...
  Loading dataset from: data/amazon-meta.txt
  Dataset loaded in 18.36 seconds
  Products processed: 542684
  Connections established: 1788725

  Dataset Statistics:
  - Products: 715483
  - Connections: 1788725
  - Avg connections per product: 2.50

  [2/3] Identifying top products...

  Top 5 Best-Selling Products:
  1. Patterns of Preaching: A Sermon Sampler
     - ASIN: 0827229534
     - Category: Book
     - Sales Rank: 396585
  2. Witness of Preaching
     - ASIN: 0804215715
     - Category: Book
     - Sales Rank: 93405
  3. The Preaching Life
     - ASIN: 156101074X
     - Category: Book
     - Sales Rank: 92111
  4. The Four Pages of the Sermon: A Guide to Biblical Preaching
     - ASIN: 0687023955
     - Category: Book
     - Sales Rank: 64877
  5. Performing the Word: Preaching As Theatre
     - ASIN: 0687074231
     - Category: Book
     - Sales Rank: 235453
```



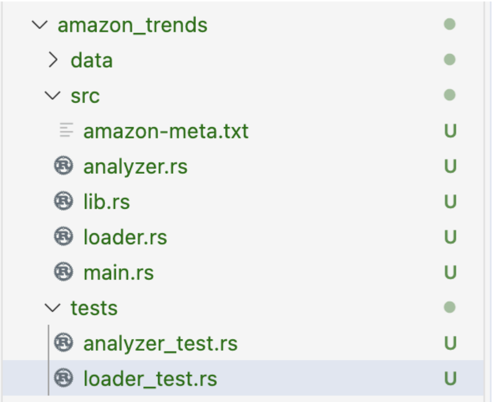
```
[3/3] Detecting market trends...

Emerging Product Trends:
Trend Group 1 (5 products):
Sample Products:
- Death in a Hot Flash (Bel Barrett Mysteries (Avon Books)) (Rank: 55423)
- Out of Hormone's Way : A Bel Barrett Mystery (Bel Barrett Mysteries (Avon Books)) (Rank: 453526)
- Mood Swings to Murder: A Bel Barrett Mystery (Rank: 272880)
Trend Group 2 (6 products):
Sample Products:
- Fresh Aire 8 (Rank: 46617)
- Fresh Aire 7 (Rank: 19860)
- Fresh Aire VI (Rank: 86255)
Trend Group 3 (7 products):
Sample Products:
- CBS Dinner Classics: Romance (Rank: 68226)
- Breakfast in Bed (Rank: 118916)
- Dinner Classics: The Sunday Brunch Album (Rank: 106364)

[4/4] Analyzing competition...

Best Market Opportunities:
1. The War of the Worlds (Opportunity Score: 0.20)
   - Current Rank: 1
   - Category: Video
2. Shirley Valentine (Opportunity Score: 0.40)
   - Current Rank: 2
   - Category: Video
3. Leslie Sansone - Walk Away the Pounds - Super Fat Burning (Opportunity Score: 1.20)
   - Current Rank: 6
   - Category: Video
4. Robin Hood - Men in Tights (Opportunity Score: 1.40)
   - Current Rank: 7
   - Category: Video
5. Richard Simmons - Sweatin' to the Oldies (Opportunity Score: 1.60)
   - Current Rank: 8
   - Category: Video
```

Interpretation:

The Amazon Product Trend Analyzer processed over 540,000 products and 1.7 million connections, revealing key insights. Preaching books, led by Patterns of Preaching, top the best-seller list, while mystery series and music collections are rising in popularity. The Video category shows strong market potential, with The War of the Worlds ranked first in sales and facing minimal competition (score 0.20). An average of 2.5 connections per product highlights a well-linked network that surfaces both popular items and hidden opportunities.

F. Usage Instructions

- Rust has to be set up
- Create the following structure of files and place the codes as given in the github repository.



```
amazon_trends              ●
  data                     ●
  src                      ●
    amazon-meta.txt        U
    analyzer.rs            U
    lib.rs                 U
    loader.rs              U
    main.rs                U
  tests                    ●
    analyzer_test.rs       U
    loader_test.rs         U
```

- Place the dataset (amazon-meta.txt) in the data folder - Because it is a big file I have attached it as a google drive link.

  Link: https://drive.google.com/file/d/1EazT7Wm5pIUqtRyxLGeLo8E_2dAkq3r6/view?usp=drive_link

- Build using the cargo run --release for maximum optimization
- Run all tests using cargo test
- It takes 20-25 seconds to run the whole code through cargo run --release

G. Usage Instructions

I used AI to help write code comments and improve the market opportunity scoring algorithm. The AI suggested optimizations for analyzing product clusters and identifying high-potential items. This sped up development while keeping the code clean and efficient.