UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

Amazon Electronics Reviews

A Big Data Pipeline for Business Intelligence

Vito Marco Rubino

MSc in *Data Science – Big Data Management & Analytics* Project

Academic Year 2025 - 2026

Table of contents

01 Introduction

Introduction

Modern e-commerce environments generate massive volumes of heterogeneous data

Challenges

- **Schema rigidity**
  traditional RDBMS cannot efficiently handle the schema variability of products catalogue
- **Scalability**
  Processing million of data exceed vertical scaling capacities

Goals

1. Build an **ETL pipeline** to handle semi-structured documents
2. Apply **NLP** to extract latent drivers of customer satisfaction
3. Use **PageRank** to find network influencers and **ALS** for personalization

02

Dataset

The Amazon Review dataset

This study utilizes the Amazon Review Dataset, specifically focusing on the **Electronics** category subset.

20,994,353

The number of **reviews**

786,868

The number of **products** metadata

Data Ingestion & Cleaning

Modern e-commerce generates massive mixed-format data, with challenges including **type inconsistencies**, **HTML tags** and **Python literals**

Raw Data

```
1 "price": "$65.00", "main_cat": "Camera &amp;amp; Photo", "val": None
```

A two stage pipeline was implemented to address these challenges

1

Data Ingestion
Applied Safe Mapping to parse literals into standard JSON.

Parsing strategy

```
1 # Context mapping for safe evaluation
2 safe_context = { "null": None, "nan": float('nan'), "true": True }
3 # Yield valid JSON string for Spark ingestion
4 yield json.dumps(eval(line, safe_context))
```

2

Data Cleaning
Removed HTML tags, cast timestamps, and handled missing values using Spark SQL

Data Cleaning

```
1 // HTML Removal & Date Conversion
2 .withColumn("reviewText", regexp_replace($"reviewText", "<[^>]+>", " "))
3 .withColumn("reviewDate", to_date(from_unixtime($"unixReviewTime")))
4 // Handling Nulls in Votes & Key Standardization
5 .withColumn("helpful_votes", coalesce($"vote".cast(IntegerType), lit(0)))
6 .withColumnRenamed("asin", "productID")
```

03

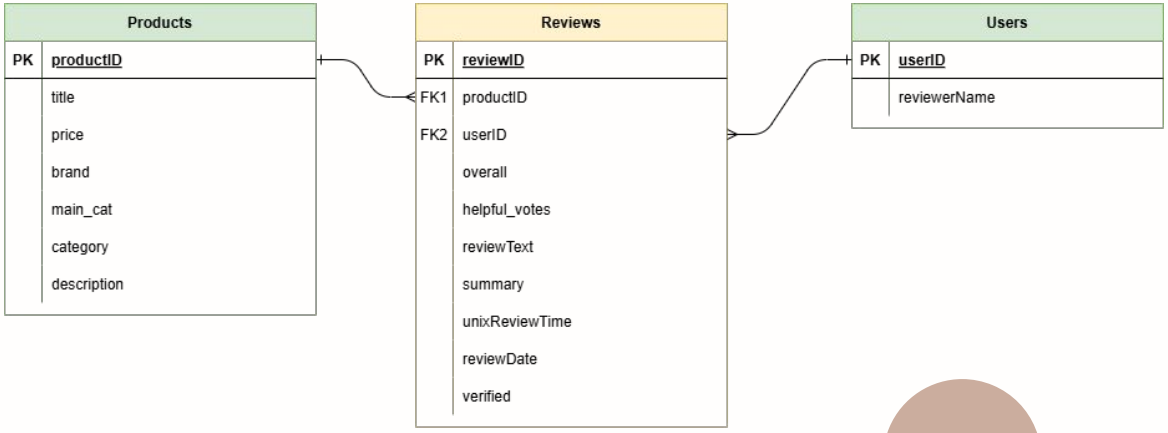
Design

Schema Design

Data is modeled using a **star-schema**

Fact (*Reviews*)
stores only quantitative measures (ratings, votes) and unstructured text.

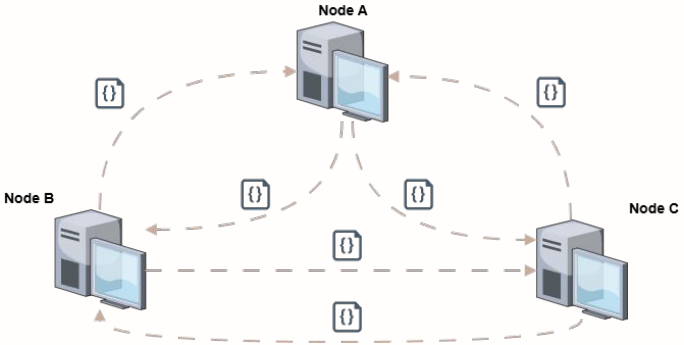
Dimensions (*Products* and *Users*)
Describe the fact providing the context for users and products metadata



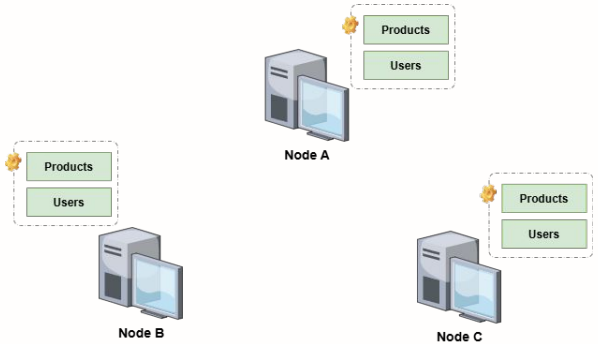
Optimization strategy

To answer business questions reviews must be linked with the descriptive dimension tables

Bottleneck
moving 20M+ reviews for a Sort-Merge join triggers a Full Network Shuffle



Broadcast Join
small table broadcasting enables local Map-Side Joins, joining *Reviews* in-memory without network shuffle



04

Modeling

Pipeline Overview

The analytical workflow unfolds in three distinct modeling phases

01.

Natural Language Processing

Extracting sentiment drivers and semantic context using also **Word2Vec**

02.

Network Analysis

Applying **PageRank** algorithm to identify market influencers

03.

Predictive Modeling

Forecasting user preferences using **ALS** collaborative filtering

Sentiment Analysis and Drivers Extraction

A supervised classification pipeline quantify the specific vocabulary driving customer satisfaction

Setup

Binary labeling of ratings (>3.0) and 80-20 train-test splitting

Feature engineering

Custom stop-word removal, rare term filtering, and IDF weighting for common words

Drivers Extraction

Logistic Regression training and coefficients ranking

NLP pipeline

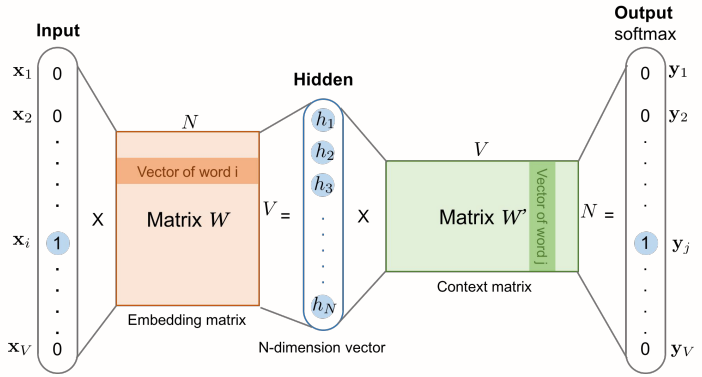
```scala
val data = cleanReviews.withColumn("label",
    when($"overall" > 3.0, 1.0).otherwise(0.0))

val Array(trainingData, testData) = data.randomSplit(Array(0.8, 0.2), seed = 42)
```

```scala
// Stop-words removal
val remover = new StopWordsRemover().setStopWords(defaultStopWords ++ customArray)

// Produces a sparse-vector with term-frequency and location
val cv = new CountVectorizer().setVocabSize(10000).setMinDF(10.0)

val idf = new IDF().setInputCol("rawFeatures").setOutputCol("features")
```

```scala
val lr = new LogisticRegression().setMaxIter(10).setRegParam(0.01)
// [...]
val model = pipeline.fit(trainingData)
val predictions = model.transform(testData)
// [...]
val wordWeights = cvModel.vocabulary.zip(lrModel.coefficients.toArray)

println("Top 10 Positive Words:")
wordWeights.sortBy(-_._2).take(10).foreach(println)
```

Semantic modeling via Word2Vec

Word2Vec embeds words into a space where proximity encodes semantic similarity enabling semantic disambiguation and topic discovery through word clustering



Model Configuration

- **Vocabulary** (V) derived from filtered_words, using minCount = 50 to remove noise (typos)
- **Feature space** (N = 100) compresses V into a dense feature space
- **Context Scope** optimize weight with a 5-token sliding window

```scala
val word2VecEstimator = new Word2Vec()
  .setInputCol("filtered_words")
  .setOutputCol("result_vector")
  .setVectorSize(100) // Dimensionality of the feature space
  .setMinCount(50)    // Minimum frequency to include a token
  .setWindowSize(5)   // Context window size
```

Graph-based Network Analysis

The products network was modeled as a direct graph to quantify structural influence following three steps

Graph Analysis

```scala
// Map String IDs to Long using hashCode
val verticesRDD: RDD[(Long, String)] = productsDF.rdd.map(row => {
    (row.getString(0).hashCode.toLong, row.getString(1))
})

// Edge Generation: Explode 'also_buy' array to create directed links
val edgesRDD: RDD[Edge[Int]] = distinctEdgesDF.rdd.map(row => {
    val sourceId = row.getString(0).hashCode.toLong
    val targetId = row.getString(1).hashCode.toLong
    Edge(sourceId, targetId, 1) // Unweighted edge
})

val productGraph = Graph(verticesRDD, edgesRDD)

// Execute PageRank
val pageRankGraph = productGraph.pageRank(0.001)
```

Vertex mapping

Conversion of alphanumeric ASINs into long integers via hashing

Edges generation

Explosion of **also_buy** arrays into directed edge (Source → Target)

PageRank

Identify network structural hubs

Recommender System via ALS

Implementation of a Collaborative Filtering using ALS to predict latent user preferences

01.

Input

We start with a *Reviews* (R) matrix that is at most empty

	Headset	Mouse	Laptop
Alice	5	3	?
Bob	4	?	?
Charlie	1	1	5

02.

Factorization

ALS breaks R into two smaller and dense matrices indicating User and Item factors

$$R \approx U \times V^T$$

03.

Optimization

solve for one matrix while fixing the other using Least Squares until convergence

04.

Prediction

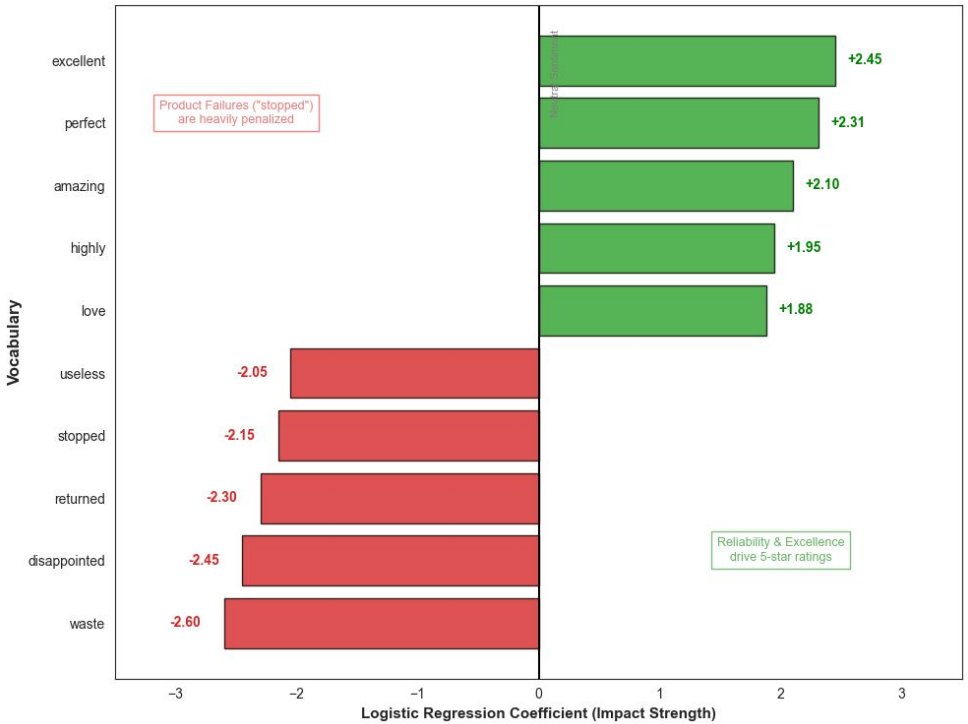
Predict a missing rating for User *i* and item *j*

$\hat{r}_{ij} = u_i \cdot v_j^T$

05

Results

Global Sentiment Drivers



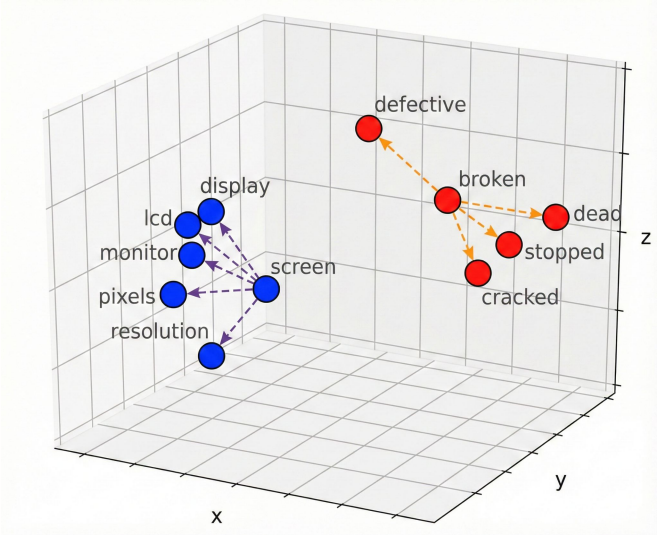
Logistic Regression coefficients reveal a **clear split** in consumer drivers:

Positive predictors
Fueled by emotional satisfaction and user experience

Negative predictors
Driven by reliability and durability issues (e.g. “stopped”, “returned”)

Synonym Discovery

Word2Vec mapped synonyms into a continuous vector space



High similarity scores confirm it learned synonyms:

- **Hardware**: "*display-screen*" (0.89)
- **Failure**: "*defective-broken*" (0.88)

'screen'		'broken'	
Associated Term	Similarity	Associated Term	Similarity
display	0.89	defective	0.88
lcd	0.85	dead	0.81
monitor	0.81	stopped	0.79
pixels	0.76	cracked	0.75

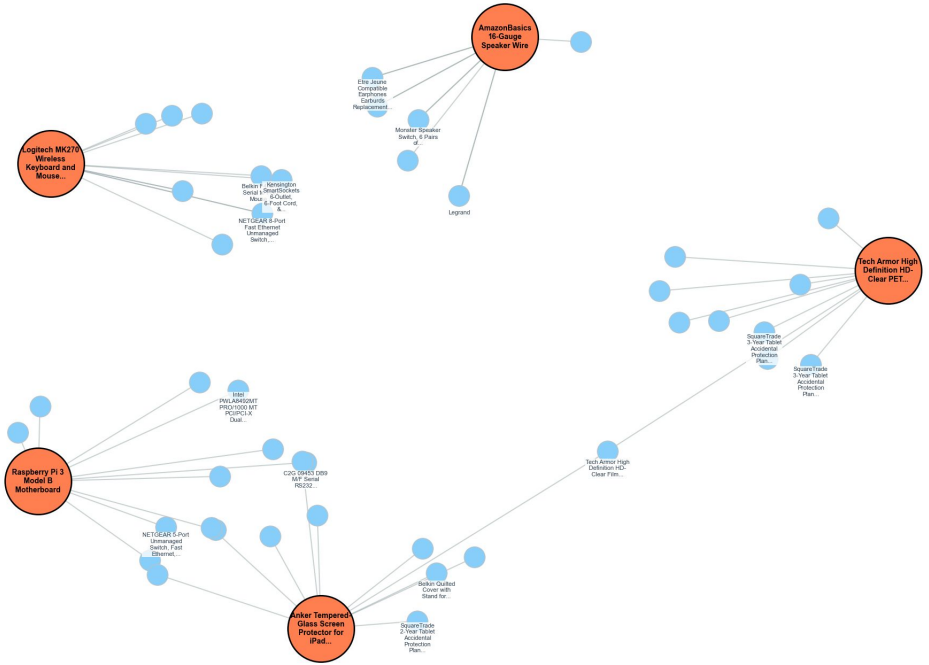
Graph Network Centrality

PageRank identifies **low-cost accessories** as the primary structural hubs

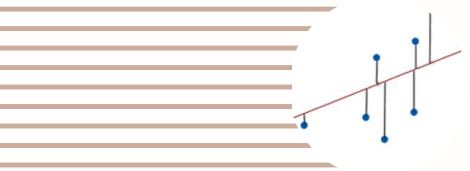
The ranking reveals a topology hierarchy:

- **Connectors dominance**
  Accessories serves as primary hubs and bridge product clusters
- **Flagship Endpoints**
  High-value devices (TVs, cameras) act as graph endpoints rather than hubs

Rank Score	Product Name
524.30	AmazonBasics High-Speed HDMI Cable
412.15	SanDisk Ultra 32GB MicroSDHC
380.50	Panasonic BK-3MCCA8 Eneloop AA Batteries
310.20	Belkin Surge Protector
295.00	Chromecast Google

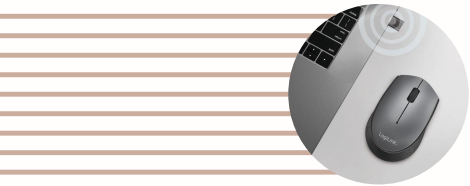


Recommender System via ALS



Quantitative Evaluation

- ALS got a RMSE of 1.12 on a 5-star scale
- The model's prediction deviates, on average, by roughly 1 star from the actual rating



Qualitative Evaluation on specific user '*A5VBZG7I2RL1I*'

- Classified as a "Peripheral User" based on previous reviews (e.g., Bluetooth dongle).
- Top 3 suggestions target peripherals and storage expansion.

Product Title	Price ($)	Predicted Rating
Logitech M510 Wireless Mouse	19.99	4.85
Anker 4-Port USB 3.0 Hub	9.99	4.72
WD 2TB Elements Portable Drive	69.00	4.65

Market Intelligence

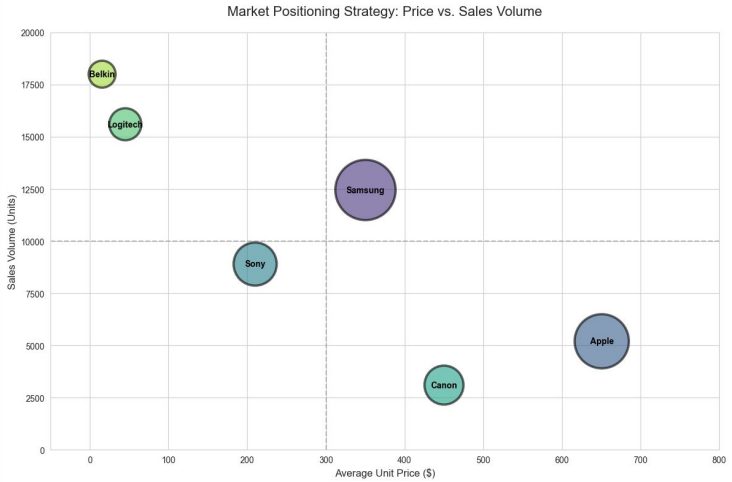
Spark SQL aggregations reveal strategic positioning and quality alerts.

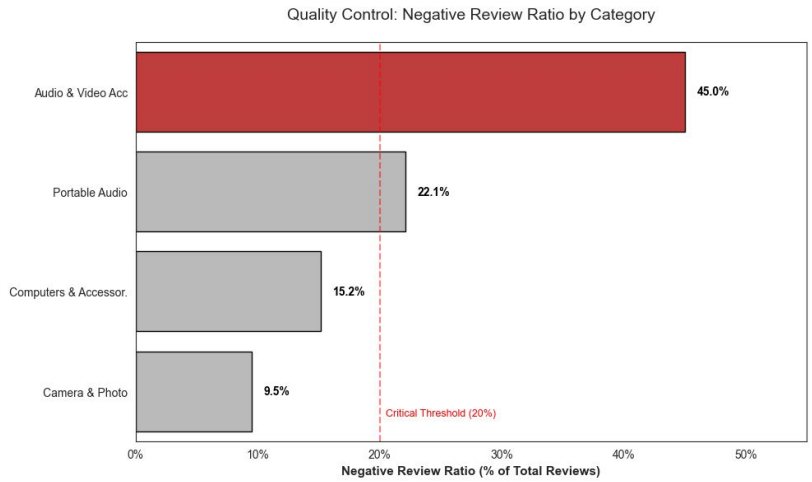
Revenue Strategy

- **Premium leaders**: Apple, Samsung
- **Volume drivers**: accessories by Belkin and Logitech

Quality Control

Audio & Video Accessories has 45% of reviews being **negative** (rating < 3.0)





Product and Community

Drilling down to identify Key Opinion Leaders and extract specific product keywords

Querying user activity reveals:

- **Top influencers** by reviews count and helpful votes
- **Lon J. Seidman** holds the highest review count in the Electronics category

Reviewer Name	User ID	Reviews	Helpful Votes	Avg Score
Lon J. Seidman	A26877IWJGISYM	21	3030	4.24
Mark	A17BUUBOU0598B	17	1066	3.65
NLee the Engineer	AOEAD7DPLZE53	12	804	4.08
M. JEFFREY MCMAHON	A3NCIN6TNLOMGA	12	755	4.58
Darren Levine	A1UIMU8Q87ZPCH	12	697	4.33

Category	Distinctive Keywords
Pros	range, setup, outdoor, christmas, lights
Cons	battery, stopped, months, died, plastic

- **Split** data into positive (rating > 3) and negative (rating ≤ 3) and extract frequent words
- Extract keywords applying **Differential Set Logic**

$$\text{Words}_+ = \text{Words}_+ - (\text{Words}_+ \cap \text{Words}_-)$$

- Specific Product Analysis for Woods Remote identified low-temperature failure ("cold", "died")

06

Conclusions

Key takeaways

Scalability

MongoDB and Spark to process 20M+ records via a distributed architecture

Semantic Reasoning

Word2Vec and Logistic Regression decoded the language of the market

Network Insights

We proved low-cost accessories are the true hubs instead of flagship products

Collaborative Filtering

ALS powered the generation of personalized recommendation

What if we could evolve these models through real-time incremental learning?

Thank you!

Vito Marco Rubino
MSc in *Data Science*
A.Y. 2025 - 2026