



ReviewLens

By: Rishabh Sheth, Kushagra
Dhall, and Joshua Dsouza

Project and Features

- Review Intelligence System that transforms raw amazon data into meaningful insights
- Robust and clean data engineering pipeline with a fully normalized SQL schema
- Advanced NLP and ML processing including:
 - Sentiment Scoring
 - Phrase extraction
 - Keyword ranking
 - Summarization
- Insight Dashboard to provide clean visualization of product summaries and sentiment trends

```
class ReviewProcessor:
    """Process reviews for sentiment, keywords, and summaries."""

    def __init__(self, db_path='reviews.db', use_gpu=False):
        self.db_path = db_path
        self.sia = SentimentIntensityAnalyzer()
        self.stop_words = set(stopwords.words('english'))
        # Add common words to stopwords
        self.stop_words.update(['product', 'item', 'amazon', 'purchase', 'buy', 'bought'])

        # Initialize spaCy for phrase extraction
        self.nlp = None
        if SPACY_AVAILABLE:
            try:
                print("Loading spaCy model...")
                self.nlp = spacy.load("en_core_web_sm")
                print("spaCy model loaded successfully.")
            except OSError:
                print("Warning: spaCy model 'en_core_web_sm' not found.")
                print("Install with: python -m spacy download en_core_web_sm")
                self.nlp = None
        else:
            print("Warning: spaCy not available. Using fallback phrase extraction.")

        # Initialize summarization model (use CPU by default, can use GPU if available)
        print("Loading summarization model...")
        device = 0 if use_gpu and torch.cuda.is_available() else -1
        self.summarizer = None

        # Try smaller models first for better compatibility
        models_to_try = [
            ("t5-small", "t5-small"),
            ("facebook/bart-large-cnn", "facebook/bart-large-cnn"),
            ("google/pegasus-xsum", "google/pegasus-xsum")
        ]

class ReviewProcessor:
    """Process reviews for sentiment, keywords, and summaries."""

    def __init__(self, db_path='reviews.db', use_gpu=False):
        self.db_path = db_path
        self.sia = SentimentIntensityAnalyzer()
        self.stop_words = set(stopwords.words('english'))
        # Add common words to stopwords
        self.stop_words.update(['product', 'item', 'amazon', 'purchase', 'buy', 'bought'])

        # Initialize spaCy for phrase extraction
        self.nlp = None
        if SPACY_AVAILABLE:
            try:
                print("Loading spaCy model...")
                self.nlp = spacy.load("en_core_web_sm")
                print("spaCy model loaded successfully.")
            except OSError:
                print("Warning: spaCy model 'en_core_web_sm' not found.")
                print("Install with: python -m spacy download en_core_web_sm")
                self.nlp = None
        else:
            print("Warning: spaCy not available. Using fallback phrase extraction.")

        # Initialize summarization model (use CPU by default, can use GPU if available)
        print("Loading summarization model...")
        device = 0 if use_gpu and torch.cuda.is_available() else -1
        self.summarizer = None

        # Try smaller models first for better compatibility
        models_to_try = [
            ("t5-small", "t5-small"),
            ("facebook/bart-large-cnn", "facebook/bart-large-cnn"),
            ("google/pegasus-xsum", "google/pegasus-xsum")
        ]

    def get_connection(self):
        pass

    def clean_text(self, text: str) -> str:
        pass

    def compute_sentiment(self, text: str) -> tuple(float, str):
        pass

    def extract_phrases_spacy(self, text: str) -> list[tuple[str, str]]:
        pass

    def extract_phrases_fallback(self, text: str) -> list[tuple[str, str]]:
        pass

    def extract_keywords(self, texts: list[str], sentiment_target: str, product_name: str = "", top_n: int = 5) -> list[tuple[str, float]]:
        pass

    def remove_overlapping_keywords_scored(self, scored_phrases: list[tuple[str, float, float, int]] -> list[tuple[str, float, float, int]]:
        pass

    def generate_insight_summary(self, texts: list[str], positive_keywords: list[tuple[str, float]], neg_themes: list[str], neg_themes: list[str]) -> str:
        pass

    def extract_themes_from_keywords(self, keywords: list[tuple[str, float]]) -> list[str]:
        pass

    def get_representative_sentences(self, texts: list[str], pos_keywords: list[tuple[str, float]], neg_themes: list[str], neg_themes: list[str]) -> str:
        pass

    def build_structured_input(self, pos_themes: list[str], neg_themes: list[str], neg_themes: list[str]) -> str:
        pass

    def clean_summary_text(self, summary: str, pos_themes: list[str], neg_themes: list[str]) -> str:
        pass

    def theme_based_summary(self, pos_themes: list[str], neg_themes: list[str], texts: list[str]) -> str:
        pass

    def generate_summary(self, texts: list[str], max_length: int = 150, min_length: int = 50) -> str:
        pass

    def extractive_fallback(self, text: str, num_sentences: int = 3) -> str:
        pass

    def process_all_reviews(self, batch_size: int = 100):
        pass

    def create_keywords_table(self):
        pass

    def process_products(self):
        pass
```



Data Loading

- Loads large CSV dataset consisting of 34k reviews
- Cleans and normalizes column names
- Missing values are handled safely
- Initial table is recreated
- Basic data is calculated which includes:
 - Review count
 - Number of unique products
- Command: `python load_data.py`

```
Loaded 26000 rows...
Loaded 27000 rows...
Loaded 28000 rows...
Successfully loaded 28332 rows
```

```
Database Stats:
  Total reviews: 28332
  Unique products: 65
```

```
Normalizing into new tables...
Normalizing...
0/28332 normalized...
5000/28332 normalized...
10000/28332 normalized...
15000/28332 normalized...
20000/28332 normalized...
25000/28332 normalized...
Normalization complete!
```

```
Creating reconstructed view...
View 'reviews_reconstructed' created!
```

Here is a snippet of one of the tables we create after normalization

reviews.db

Filter 9 tables... Rows: 65 Filter 65 rows... Upgrade to PRO

TABLES		produ... # ↕	name	brand	manufacturer	manufactur...
> categories		Filter...	Filter...	Filter...	Filter...	Filter...
> images						
> product_categories						
> product_keywords						
▼ products						
ROWID						
# product_id 🔑						
name						
brand						
manufacturer						
manufacturerNumber						
> reviews						
> reviews_normalized						
> sqlite_sequence						
> users						
▼ VIEWS (EXPERIMENTAL)						
> reviews_reconstructed						

1	1	AmazonBasics AAA Performance Alkaline Batteries (36 C...	Amazonbasics	AmazonBasics	HL-002619
2	2	AmazonBasics Nylon CD/DVD Binder (400 Capacity)	Amazonbasics	AmazonBasics	YBB12400R2
3	3	Amazon Echo ,A\ White	Amazon	Amazon	B01E6AO69U
4	4	Amazon Echo Show - Black	Amazon	Amazon	B01J24C0TI
5	5	Echo Spot Pair Kit (Black)	Amazon	Amazon	B073SQYXTW
6	6	Fire TV Stick Streaming Media Player Pair Kit	Amazon	Amazon	B00ZV9RDKK
7	7	AmazonBasics AA Performance Alkaline Batteries (48 Co...	Amazonbasics	AmazonBasics	LR6G0748FFPAB-US
8	8	AmazonBasics Ventilated Adjustable Laptop Stand	Amazonbasics	AmazonBasics	278A0
9	9	AmazonBasics Backpack for Laptops up to 17-inches	Amazonbasics	AmazonBasics	NC1306167R1
10	10	AmazonBasics 11.6-Inch Laptop Sleeve	Amazonbasics	AmazonBasics	NC1303151
11	11	AmazonBasics 15.6-Inch Laptop and Tablet Bag	Amazonbasics	AmazonBasics	NC1305224R1
12	12	AmazonBasics External Hard Drive Case	Amazonbasics	AmazonBasics	HL-001725
13	13	Expanding Accordion File Folder Plastic Portable Docum...	Amazonbasics	AmazonBasics	PBH-2557
14	14	Cat Litter Box Covered Tray Kitten Extra Large Enclosed ...	Amazonbasics	AmazonBasics	5122-S
15	15	Amazon 9W PowerFast Official OEM USB Charger and P...	Amazon	Amazon	55-000662
16	16	Kindle PowerFast International Charging Kit (for accelera...	Amazon	Amazon Digital Services	53-000148



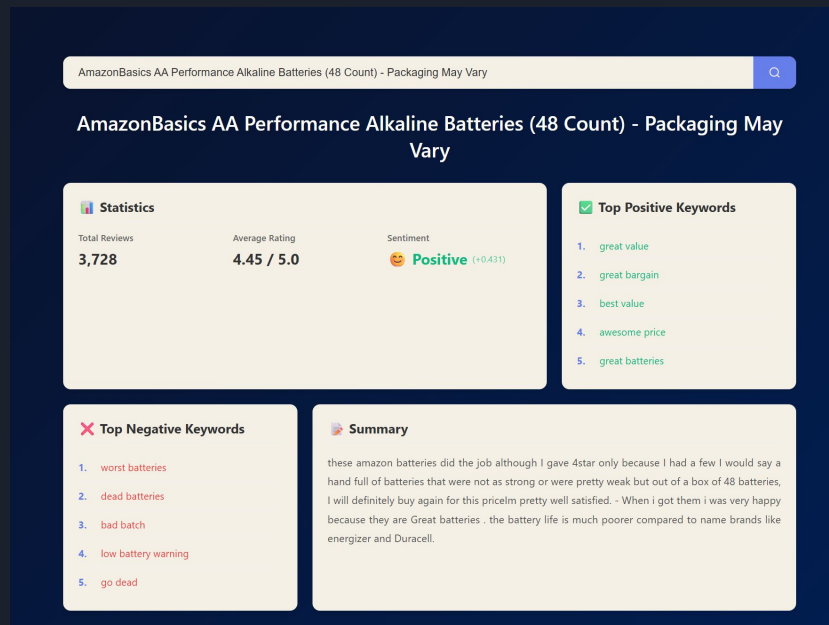
Processing Reviews + ML,NLP Pipeline

- Run this command “python process_reviews.py”, which goes through all the reviews from the database and runs the machine learning pipeline.
- The pipeline consists of the following
 - NLTK's Sentiment Intensity Analyzer to compute a score that helps assign the labels such for positive and negative feedback.
 - spaCy to extract meaningful noun chunks and phrases from the reviews. Also used to remove generic terms like “amazon” from the reviews
 - Keyword Scoring Algorithm for this pipeline, which combines sentiment strength (65%), frequency normalization (20%), and a length based bonus metric (15%).

```
❖ (.venv) PS C:\Kushagra\Personal Projects\projects\ReviewLens> python process_reviews.py
=====
Processing Reviews: Sentiment, Keywords, and Summaries
=====
Loading spaCy model...
spaCy model loaded successfully.
Loading summarization model...
Attempting to load t5-small...
Device set to use cpu
Successfully loaded t5-small.
Processing reviews for sentiment analysis...
Found 28332 reviews to process
Processed 100/28332 reviews...
Processed 200/28332 reviews...
Processed 300/28332 reviews...
```

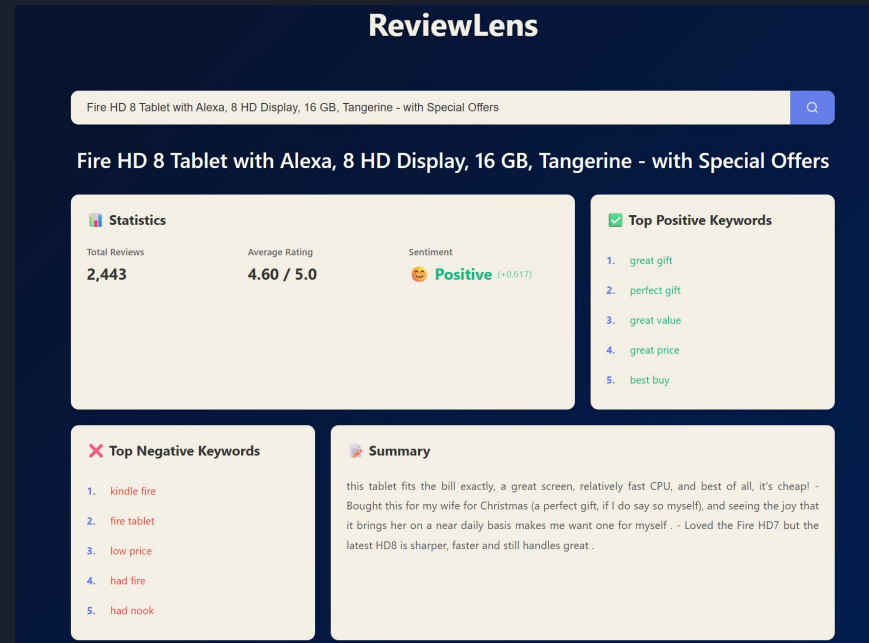
Frontend Website

- The tool correctly identifies positive themes that are seen in high frequency in the reviews including:
 - Great value
 - Best value
 - Awesome price
- Negative keywords, for instance, dead batteries, bad batch, worst batteries, etc., accurately reflect the complaints that customers have had about the product
- The sentiment score aligns with the variety of reviews (in this case positive overall)
- The summary presented captures recurring opinions of customers in the reviews.



Frontend Website (cont.)

- The tool also had errors with a few products like the one presented here
- Some extracted keywords, for example, kindle fire and fire tablet, are names of the product and are not negative sentiments
- The sentiment analyzer misinterprets the word fire as a negative indicator because of its literal meaning and doesn't understand that is the name of the product
- Misinterpretation of the product's name in the reviews throws the analyzer off which then results in inaccurate results, misleading customers



What would we enhance?

- Using a better sentiment analyzer model that stays aware of entities like Kindle Fire, Fire HD, etc.
- Improving keyword classification to avoid false negatives and positives
- Quality checking generated summaries to prevent irrelevant and repetitive text
- Expanding the database to keep receiving new reviews that get posted and update the sentiment scores and summaries
- Implement a filtration tool to filter certain product categories and focus on sentiment scores for specific features

