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

```python
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")
        ]
```

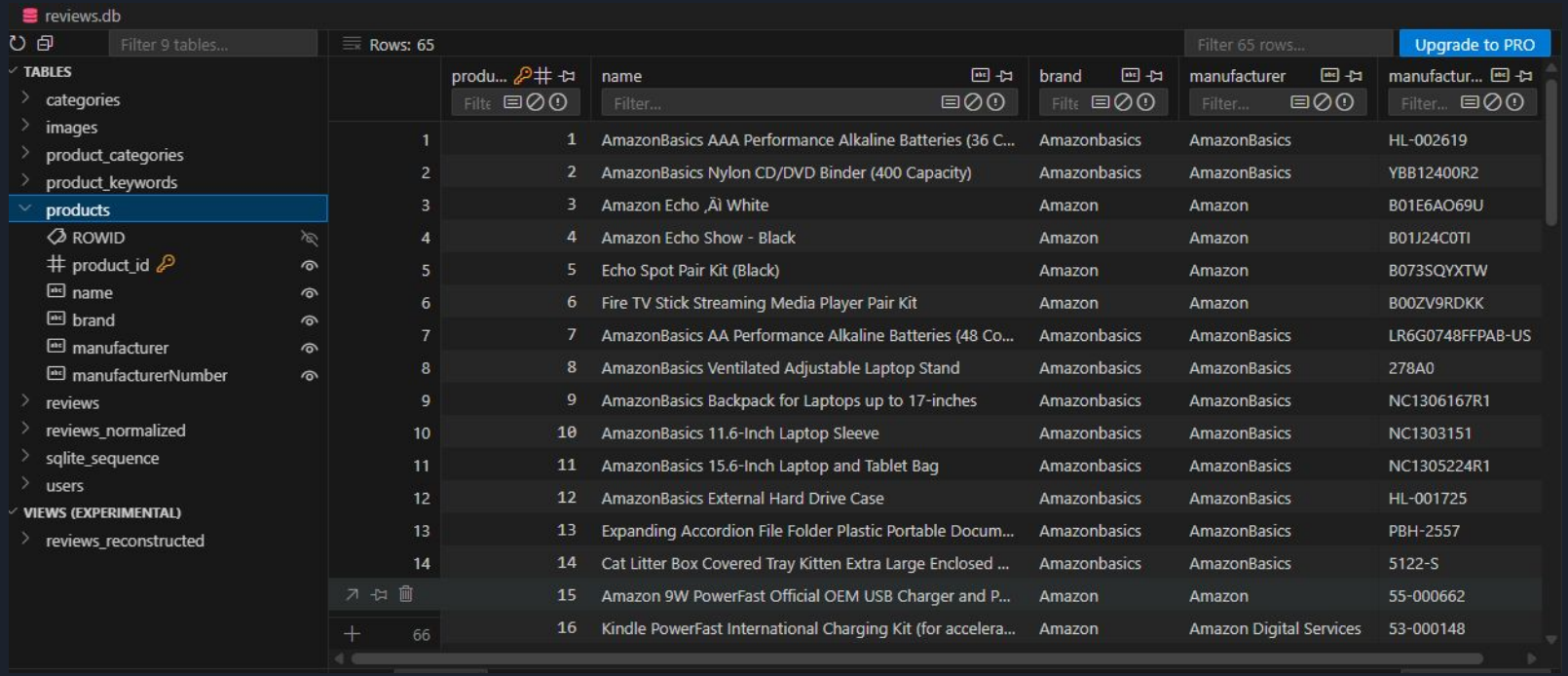
```python
class ReviewProcessor:
    """Process reviews for sentiment, keywords, and summaries."""

    def __init__(self, db_path='reviews.db', use_gpu=False): ...

    def get_connection(self): ...

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

    def compute_sentiment(self, text: str) -> Tuple[float, str]: ...

    def _extract_phrases_spacy(self, text: str) -> List[Tuple[str, str]]: ...

    def _extract_phrases_fallback(self, text: str) -> List[Tuple[str, str]]: ...

    def extract_keywords(self, texts: List[str], sentiment_target: str, product_name: str = "", top_n: int = 5) -> List[Tuple[str, float]]: ...

    def _remove_overlapping_keywords_scored(self, scored_phrases: List[Tuple[str, float, float, int]]) -> List[Tuple[str, float, float, int]]: ...

    def _remove_overlapping_keywords(self, keywords: List[Tuple[str, int]]) -> List[Tuple[str, int]]: ...

    def generate_insight_summary(self, texts: List[str], positive_keywords: List[Tuple[str, float]], ...

    def _extract_themes_from_keywords(self, keywords: List[Tuple[str, float]]) -> List[str]: ...

    def _get_representative_sentences(self, texts: List[str], pos_keywords: List[Tuple[str, float]], ...

    def _build_structured_input(self, pos_themes: List[str], neg_themes: List[str], ...

    def _clean_summary_text(self, summary: str, pos_themes: List[str], neg_themes: List[str]) -> str: ...

    def _theme_based_summary(self, pos_themes: List[str], neg_themes: List[str], texts: List[str]) -> str: ...

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

    def _extractive_fallback(self, text: str, num_sentences: int = 3) -> str: ...

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

    def create_keywords_table(self): ...

    def process_products(self): ...
```

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

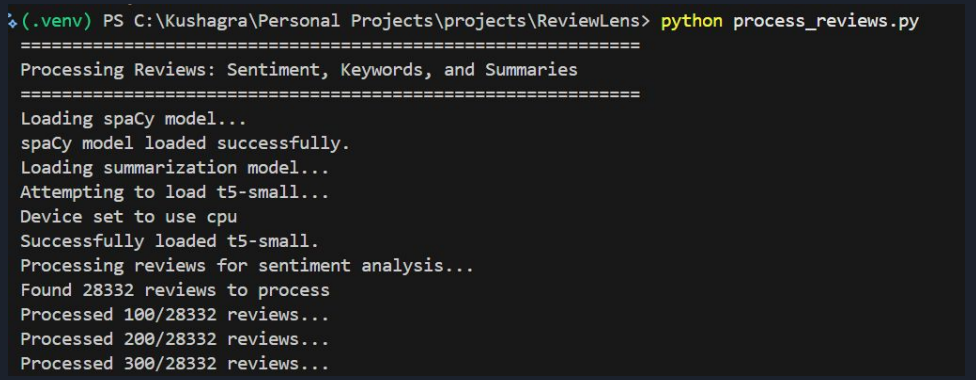
```
Loaded 26000 rows...
Loaded 27000 rows...
Loaded 28000 rows...
Successfully loaded 28332 rows

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



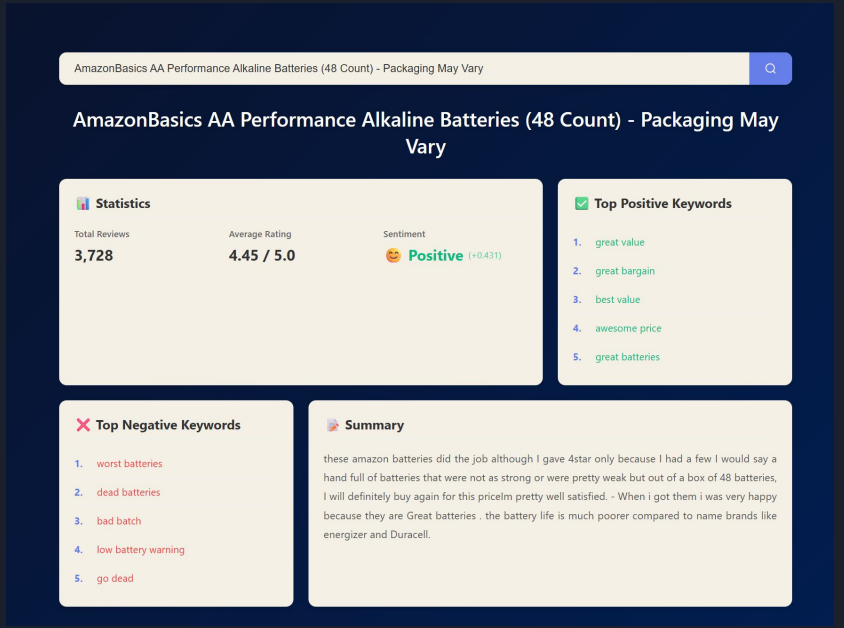
Processing Reviews + ML,NLP Pipeline

- 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%).



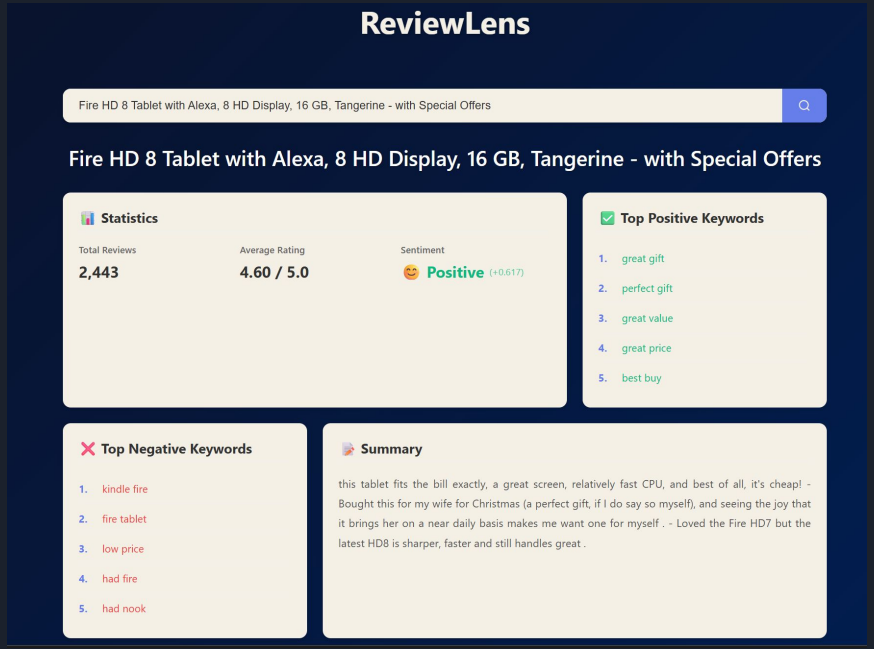
Frontend Website

- 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