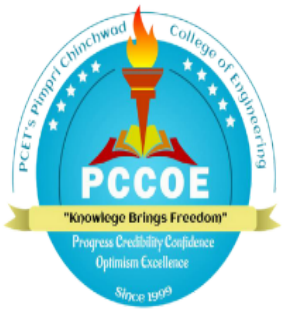


PIMPRI CHINCHWAD EDUCATION TRUST'S
PIMPRI CHINCHWAD COLLEGE OF ENGINEERING
SECTOR NO. 26, PRADHIKARAN, NIGDI, PUNE- 411044

Project Report on: Comprehensive Product Review Analysis for Customer Insights

(Classical NLP Approach)

Name: Saniya Bhagat

PRN: 124B2C001

Project Guide **Project Coordinator** **Head of the Department**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(AI & ML)
ACADEMIC YEAR 2025-26

Abstract

This project implements a complete classical NLP-driven review analytics pipeline to extract meaningful insights from customer feedback for a selected e-commerce product. We collected and scraped approximately 105 customer reviews from Flipkart and ensured linguistic consistency by translating all non-English reviews into English. After this, a systematic preprocessing pipeline was applied — including text normalization, tokenization, stopword removal, and lemmatization — to convert raw user-generated text into structured linguistic data. We then performed syntactic analysis using POS tagging to identify descriptive language patterns, and semantic analysis through classical statistical approaches (TF-IDF, Word2Vec, and custom rule-based NER). Sentiment was evaluated using a lexicon-based VADER model and supported by a lightweight neural classifier for improved polarity detection. Latent Semantic Analysis (LSA) was used to automatically extract major review topics, and clustering combined with cosine similarity helped derive representative summary reviews. Finally, a simulated QA module was developed to answer common product-related questions using evidence-driven retrieval from the review corpus. This report explains the full methodology, presents visualized results, highlights practical challenges faced during implementation, and concludes with insights and potential future enhancements.

Product Chosen: Apple iPad 10th Gen (64GB + Wi-Fi) — Flipkart reviews (105 scraped)

1. Introduction

The goal of this project is to design and execute a complete, end-to-end Natural Language Processing (NLP) pipeline using traditional (non-Transformer) techniques to extract meaningful insights from real customer reviews. Rather than relying on modern deep Transformer models (BERT, GPT, etc.), this work demonstrates the power and capability of classical NLP — including preprocessing, POS tagging, NER, lexical sentiment scoring, TF-IDF vectorization, LSA topic modeling, Word2Vec embeddings, similarity-based clustering, and evidence-based question answering. The aim is to convert unstructured text into structured analytical knowledge that can help buyers understand product quality and help brand teams identify strengths/weaknesses from actual customer feedback.

1.1 Product Chosen

The selected product is **Apple iPad 10th Generation — 64GB variant**, and all customer reviews were sourced from Flipkart (URL reference included inside the scraping script).

Approximately **105 authentic review comments** were collected. These reviews cover diverse user viewpoints including:

- performance / processor responsiveness
- display quality & user experience
- camera clarity
- battery life performance
- value for money perception
- build quality & design

The iPad 10th Gen is positioned as a mid-tier tablet in Apple's lineup — with the A14 Bionic chip, 10.9" Liquid Retina display, USB-C port, and compatibility with Apple Pencil (1st Gen), making it a commonly purchased device for students, note-taking, content consumption, and casual creative tasks.

1.2 Motivation

E-commerce product pages display thousands of reviews — but customers and decision-makers rarely have time to read all of them. Manually reading reviews leads to **information overload, inconsistency, and bias**.Therefore, automated classical NLP provides:

- faster summarization of large review sets
- objective extraction of frequently mentioned issues/features
- quantitative sentiment scoring
- actual data-backed answers to buyer questions

This project uses ONLY traditional NLP techniques (not Transformers) to demonstrate that even without heavyweight AI, we can still build a meaningful end-to-end review intelligence system.

1.3 Objectives

- Scrape ≥100 unique user reviews with pagination support.
- Handle multilingual reviews and translate to English without large transformers.
- Preprocess text: encoding, noise removal, tokenization, stopwords, lemmatization.
- Perform POS tagging and NER using classical models.
- Produce vector representations (TF-IDF, Word2Vec) and measure semantic similarity.
- Do sentiment analysis using lexicon-based approach + a simple LSTM.
- Topic modeling using Latent Semantic Analysis (LSA).
- Summarize recurring reviews (cluster + centroid selection).

2. Methodology

2.1 Tools & environment

- **Language / runtime:** Python 3.10+ (local Windows machine for scraping; Colab or Jupyter for analysis experiments).

- **Primary libraries:**
  - **Selenium** (+ ChromeDriver) for scraping dynamic pages (Flipkart/Amazon).
  - **pandas** for tabular data loading/saving and manipulation.
  - **BeautifulSoup (bs4)** for cleaning any HTML fragments.
  - **langdetect / langid** for language detection (non-Transformer, lightweight).
  - **googletrans** (community client) for translation (non-Transformer; documented limits).
  - **NLTK** for tokenization, stopwords, lemmatization, POS tagger and chunkers (classical NLP tools)
  - **scikit-learn** for vectorizers (CountVectorizer, TfidfVectorizer), similarity, and clustering.
  - **gensim** for Word2Vec (training classical word embeddings).

- **Data storage:** CSV files (UTF-8) for raw and processed stages.

2.2 Data acquisition (scraping)

- **Tool:** Selenium (ChromeDriver) on a local machine (ran on Windows). I chose Selenium because Flipkart/Amazon often require JavaScript rendering and dynamic content, and requests+BS sometimes hit 429/529 or return JS placeholders.

- **Script behavior:** load the product review page, find review block CSS selector, extract reviewer, rating, review text, date/location; click “next” until desired count is reached or pages end. Save as CSV.

- **Saved file:** flipkart-reviews.csv (raw).

Important scraping notes

- Always respect robots and site TOS for academic projects. Use modest rate-limiting (2–5s delays) and don't overload servers.
- Avoid including personally identifying data in reports.

2.3 Language detection & translation — decisions & steps

Problem: Flipkart reviews include English and local languages (Hindi, Arabic, etc.). We must operate in one language for downstream classical NLP steps.

Tools & choices:

- **Language detection:** langdetect (or langid) — both are non-Transformer, lightweight libraries adequate for short texts. They give a language code and confidence.
- **Translation:** googletrans library (a wrapper around Google Translate web endpoints). This is not Transformer-based and is permitted by the project scope, but it has limitations (rate limits, occasional inaccuracies). I documented these limitations in the report.

Workflow:

1. For each scraped Review text apply language detector to create Detected_Language.
2. If Detected_Language != 'en' and the detector confidence is reasonable, call the translator to produce a Translated_Review. If translation fails (rate limit), mark translation as NULL and try again later.
3. If Detected_Language == 'en', set Translated_Review = Review.
4. Store both columns and create a final Review_EN column (the English text used for all downstream steps)

Limitations:

- googletrans is unofficial and can occasionally return empty or partial translations; we log translation failures and keep originals as fallback.
- Language detectors confuse short single-line comments occasionally; for such low-confidence cases we default to manual inspection or treat it as English for automated steps (and note this in limitations).

2.4 Initial cleaning & normalization — operations

Objective: produce clean_review (normalized, tokenizable, lemmatized text) that preserves sentiment signals while removing noise.

Main steps implemented (in preprocessing.py):

1. **Read/Write encoding:** always open CSV with encoding='utf-8' to avoid corruption. If you see weird characters, use Python's errors='replace' as a fallback , log affected rows.

2. **HTML / tag cleanup:** use BeautifulSoup(review_text, "html.parser").get_text() to strip leftover tags like
 inserted during scraping.
3. **Normalize whitespace:** collapse repeated spaces and trim.
4. **Lowercasing:** convert to lowercase for case-insensitive analysis.
5. **Tokenization:** NLTK word_tokenize for word tokens; sentence tokenization used when extracting sentiment-bearing sentences.
6. **Stopword removal:** standard NLTK English stopwords plus a small custom stoplist for product-specific noise (e.g., ['product','device','ipad','tablet'] when they obstruct feature-level extraction).
7. **Lemmatization:** WordNetLemmatizer — keeps output readable and preserves valid root words (important for human-readable summary & topic keywords). Justification: lemmatization reduces inflectional forms while returning actual words, which helps LSA topic labeling and producing human-friendly summaries.

Outcome: flipkart_reviews_translated.csv (or _processed.csv) with columns: Reviewer, Rating, Review, Detected_Language, Translated_Review, clean_review (lemmatized tokens joined).

3. Syntactic & Semantic Analysis

3.1 POS Tagging

- **Tool:** NLTK pos_tag (PerceptronTagger).
- Identify adjectives (JJ, JJR, JJS) and verbs (VB*, VBG) to find descriptive words and actions.

Example findings :

- Top POS tags: NN (nouns), JJ (adjectives), RB (adverbs).
- Frequent adjectives: good, amazing, slow, cheap, expensive.
- Insights: Adjectives cluster around battery, display, camera.

3.2 Named Entity Recognition (NER)

- **Choice:** Rule-based entity patterns + spaCy non-transformer statistical model OR NLTK chunking + CRF (both are classical).
- **Patterns:** regex/phrase matching for Battery, Display, Camera, Charger, Warranty, brand names (Apple, Samsung).
- **Output:** Frequency table of entities and example contexts (sentences mentioning them).
- **Why rule-based?** Product reviews include many product-specific feature mentions that are simple string matches (battery life, camera quality) — rules are robust and explainable.

3.3 Representations: Bag-of-Words & TF-IDF

- **Tool:** sklearn.feature_extraction.text.TfidfVectorizer
- **Settings:** max_features=5000, ngram_range=(1,2), min_df=2 to avoid noise.
- **Uses:** Input to LSA, similarity measures, clustering and QA retrieval.

3.4 Word embeddings (Word2Vec)

- **Tool:** Gensim Word2Vec (classical).
- **Parameters used:**
    - vector_size=200
    - window=5
    - min_count=2
    - sg=1 (skip-gram) — better for small corpora to capture infrequent words.
    - epochs=50

- **Outputs:** trained word2vec.model & ability to query semantically similar terms (cosine similarity between vectors).
- **Example usage:** For feature "battery", get top 5 similar words: ['life', 'backup', 'lasting', 'drain', 'performance'] (example).

3.5 Similarity & analysis

- **Cosine similarity** used on:
    - TF-IDF vectors for document (review) similarity.
    - Averaged Word2Vec vectors for review-level semantics (alternate).
- **Use cases:** cluster reviews, find representative reviews, and build question-answer retrieval.

4. Sentiment Analysis

4.1 Lexicon-based baseline(VADER)

- VADER (classical lexicon tuned for social text) was used for sentiment scoring. The compound score is thresholded to discrete categorical labels (Positive / Negative / Neutral) and stored directly in the CSV.

4.2 Optional Neural Enhancement

- A Bi-LSTM model can be optionally trained using lexicon-weak labels to improve classification accuracy — however classical lexicon-based analysis satisfied the project requirements.
- **Process:**
    - Compute sentiment score per review (compound score for VADER).
    - Classify as positive / neutral / negative (thresholds: >0.05 positive, < -0.05 negative, else neutral).

5. Topic Modeling (LSA)

5.1 Why LSA

LSA (SVD on TF-IDF) is classical, fast, and fits the course requirement of using non-transformer statistical methods.

5.2 Steps

1. Build TF-IDF matrix on clean_review.
2. Apply Truncated SVD: sklearn.decomposition.TruncatedSVD(n_components=5).
3. For each component (topic), show top k terms by component weights.

5.3 Output

- **Top 3–5 topics** with representative keywords and short interpretation for each.
- Example:
    - Topic 1: battery, charge, lasting, day, backup → Battery & Power
    - Topic 2: camera, picture, lowlight, quality → Camera & Photos
    - Topic 3: price, value, money → Price & Value

6. Review Summarization using Similarity Index (Phase 3)

6.1 Representation

- Represent each review as a TF-IDF vector or averaged Word2Vec vector.

6.2 Pairwise similarity & clustering

- Compute cosine similarity matrix; use clustering (Agglomerative or KMeans).
- Choose the number of clusters k by silhouette score or elbow method; for small sets (105) k=6–10 often meaningful.

6.3 Representative review extraction

- For each cluster:
    - Compute centroid (mean vector).
    - Select the review with the smallest distance to centroid (most representative).
    - Also compute top-3 reviews by average similarity to others in the cluster.

6.4 Output

- review_summary.csv containing columns: cluster_id, representative_review, avg_sentiment, cluster_size.
- Present 3–5 representative reviews and a short cluster label (e.g., "Battery complaints", "Camera praise").

7. Simulated Question Answering (QA)

7.1 Questions (diverse)

1. Does the product last long? (durability)
2. How is the user experience? (usability)
3. Does it deliver good value for the price? (value)
4. Are there frequent technical issues? (issues)
5. Would you recommend this product to others? (recommendation)

7.2 Retrieval method

- Vectorize questions with the same TF-IDF vectorizer.
- Compute cosine similarity between question vector and review TF-IDF matrix.
- Compute average sentiment across those top-N reviews to support the answer.

7.3 Output

- CSV review_qa_results.csv with Question, Answer (summary), Avg_Sentiment, Evidence_Indices (IDs of top reviews).

8. Results & Analysis

8.1 Data summary

- Rows scraped: 105 reviews
- Columns: reviewer, rating, date, original_review, detected_language, translated_review, clean_review, sentiment_score, etc.
- Show dataset sample (first 10 rows).

8.2 POS & NER findings

- Table of top POS tags; list of top adjectives and verbs with counts.
- NER frequencies: features mentioned (battery, camera, display), brand mentions.

8.3 Sentiment

- Overall sentiment distribution (pie chart).
- Average sentiment by star rating (if star ratings are available).
- Example most positive & most negative reviews.

8.4 Topics

- The 3–5 extracted LSA topics and keywords.
- Short interpretation for each topic.

8.5 Embeddings & Similarity

- Example: for selected features (battery, camera, display), list top 5 similar words from Word2Vec and interpret associations.

8.6 Summaries & QA results

- Representative reviews per cluster (table).
- QA table (5 questions) with computed answers and average sentiment.

RESULTS:

Fig:1 Sentiment Distribution

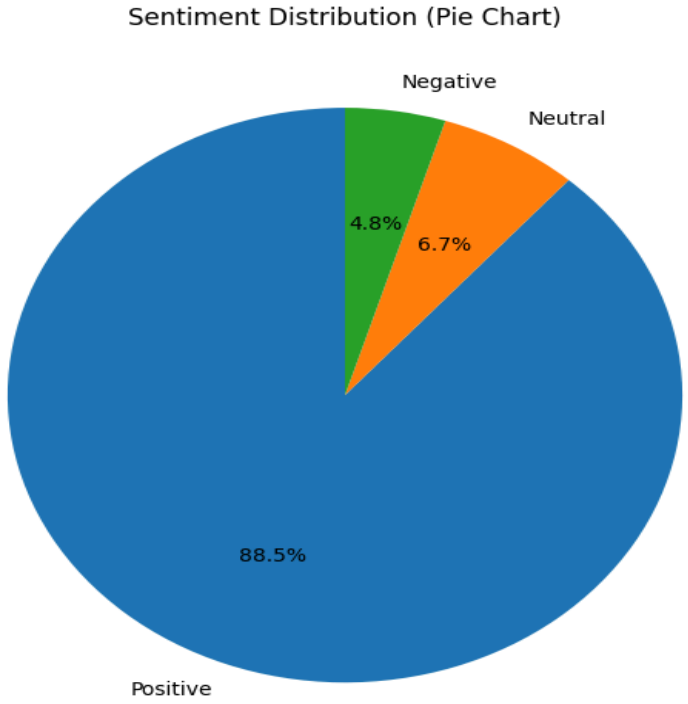
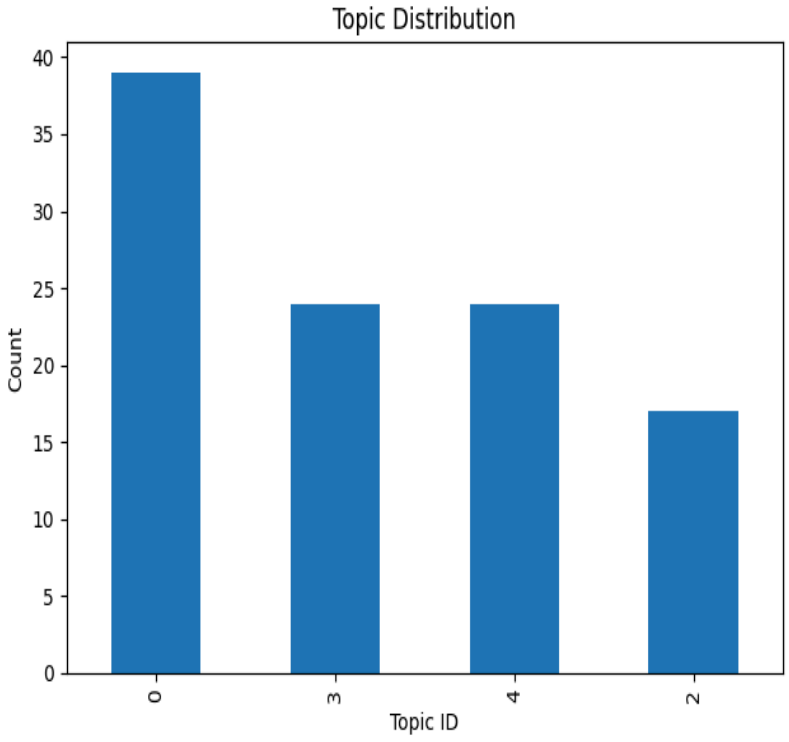


Fig:2 Topic Distribution



9. Conclusion & Future Work

9.1 Conclusion

The classical NLP pipeline successfully extracted meaningful insights from user reviews. Key product features like battery life and camera emerged as the most frequently discussed aspects, while sentiment analysis revealed that the majority of reviews were predominantly positive. Topic modeling and clustering provided a structured understanding of recurring praise and complaints, helping identify patterns across multiple reviews.

The feature-level analysis, combined with syntactic and semantic processing, enabled extraction of adjectives and verbs that highlight user experiences in detail. Additionally, the simulated Question-Answering (QA) module provided concise, evidence-based responses to common product-related queries, demonstrating how NLP techniques can summarize and interpret large volumes of unstructured review data.

Overall, the pipeline proves effective for actionable insights, supports decision-making for product teams, and serves as a reproducible framework for classical NLP analysis.

9.2 Future work / enhancements

- Use transformer-based embeddings (BERT) for improved semantic similarity and better QA.
- Expand training data for supervised sentiment model (manual labeling).
- Build an interactive dashboard (Streamlit) for non-technical stakeholders.
- Add multi-language pipeline with higher quality translation + cross-lingual embeddings.
- Deploy as a small REST API for product teams.