Aspect-Based Sentiment Analysis (ABSA)

By Primac Alexandru

Table of Contents

Introduction

- Aspect-Based Sentiment Analysis (ABSA) is a fine-grained form of sentiment analysis that focuses on identifying not only the overall polarity of a text (positive, negative, or neutral) but also the sentiment toward specific *aspects* mentioned in it.

- The main goal of this project was to develop and compare three ABSA implementations, each representing a distinct methodological family:

  - **Lexicon-Based ABSA** – rule-based, interpretable, linguistically grounded

  - **Transformer-Based ABSA** – pretrained neural model (DeBERTa-v3)

  - **LLM-Based ABSA** – prompt-driven reasoning using a local large language model (DeepSeek via Ollama)

Methodology

Lexicon-Based ABSA

- This implementation combines **spaCy** for syntactic analysis and **VADER Sentiment** for lexicon-based polarity detection.
  The algorithm works in four stages:
  - **Aspect extraction**: noun chunks are identified as potential aspects (e.g., "battery life", "customer service").
  - **Opinion linking:** adjectives and verbs are matched to aspects through dependency parsing.
  - **Sentiment scoring**: the opinion phrase is analyzed with VADER, and modifiers such as "*very*", "*slightly*", or negations ("*not good*") adjust the score.
  - **Aggregation**: for multiple mentions of the same aspect, only the most confident sentiment is retained.
  - Emoji normalization and intensity adjustments were added to improve robustness for informal language.

Methodology

Transformer-Based ABSA

- The transformer approach uses the **yangheng/deberta-v3-base-absa-v1.1** model from Hugging Face.
  This model is trained specifically for ABSA and expects input in the format:
  - "text [SEP] aspect"
  - For each detected aspect, the model predicts one of three classes: *Positive*, *Negative*, or *Neutral*.
    A simple heuristic extracts candidate aspects (nouns) using regular expressions, though this could be replaced with spaCy for more advanced parsing.

- The method leverages **contextual embeddings** to handle polysemy (e.g., "light" as in "*light color*" vs "*light weight*") and multi-aspect relationships.

Methodology

LLM-Based ABSA (Ollama)

- The third implementation uses a **local large language model** via the **Ollama framework**, specifically ***DeepSeek-V3.1***.
  A structured prompt instructs the LLM to:
  - extract aspects,classify their sentiment, and output structured JSON (aspect, sentiment, confidence).

- This approach mimics the reasoning-based ABSA capability of modern generative LLMs, capable of detecting subtle irony, emotion, and context shifts.

Design Decisions

- **Unified API**
  A common abstract base class (ABSAAnalyzer) was created to define a consistent analyze(text) interface.
  Each implementation inherits from it and returns a list of standardized AspectSentiment dataclass objects.

- **Reusability and Modularity**
  Shared helpers (e.g., negation detection, result aggregation) were isolated in utils.py to avoid redundancy.

- **Comparability**
  The testing script (api_integration.py) was designed to evaluate all analyzers on identical datasets and metrics for fair benchmarking.

- **Local LLM Choice**
  Ollama was chosen for offline execution, privacy, and reproducibility — ensuring results don't rely on external APIs or network connectivity.

Experimental Setup

Dataset:

- Two datasets were used:
  - **Test Dataset (25 samples)** – quick verification of functionality (data/test_samples.json)
  - **Evaluation Dataset (108 samples) –** comprehensive benchmark across diverse domains (restaurants, electronics, education, travel, and sarcasm). (data/evaluation_)

Evaluation Metrics:

- **Aspect Accuracy –** percentage of correctly detected aspects
- **Sentiment Accuracy –** correct sentiment polarity per aspect
- **Total Predictions** – count of extracted (aspect, sentiment) pairs

Experimental Setup

Testing and Code Coverage

- Each ABSA implementation was systematically tested using Python's built-in unittest framework.
  The tests evaluated correctness, robustness, and error handling (including malformed inputs, missing aspects, and retry behavior).
- Code coverage was measured using the coverage.py tool, producing the following results:

  - **Lexicon-Based ABSA:** 79% coverage
    (Uncovered lines mostly involve rare linguistic modifiers and emoji mappings.)
  - **Transformer-Based ABSA:** 92% coverage
    (Highest coverage, including sentiment classification, multi-aspect support, and error resilience.)
  - **LLM-Based ABSA (Ollama):** 81% coverage
    (Strong coverage of retry logic, regex-based JSON recovery, and structured output conversion.)

Results & Analysis

Quantitative Results:

Evaluation Dataset (108 samples):

Implementation	Aspect Accuracy (%)	Sentiment Accuracy (%)	Total Predictions	Correct Sentiments
Lexicon-Based	64.795918	27.551020	186	54
Transformer-Based	77.040816	65.816327	692	129
LLM-Based	83.163265	77.040816	207	151

Test Dataset (25 samples):

Implementation	Aspect Accuracy (%)	Sentiment Accuracy (%)	Total Predictions	Correct Sentiments
Lexicon-Based	82.978723	53.191489	49	25
Transformer-Based	93.617021	82.978723	161	39
LLM-Based	91.489362	82.978723	47	39

Performance Metrics:

Performance Metrics

Model	Total Runtime Test Data	Total Runtime Evaluation Data	Average time per sample	Hardware Usage	Notes
Lexicon-Based	0.42s	1.75s	0.02s	CPU only	Fastest, minimal resource use
Transformer-Based	33.57s	151.44s (2.51 minutes)	~1.34s	CPU / GPU	Slower, high accuracy
LLM-Based	187.35s (3.12 minutes)	865.36 (14.42 minutes)	~8.01s	CPU+RAM heavy	Slowest, but most context-aware

Discussion

Strengths & weaknesses, challenges faced and possible use cases

- Each approach to Aspect-Based Sentiment Analysis demonstrated unique advantages and limitations, reflecting the trade-offs between interpretability, contextual understanding, and computational efficiency.

- The **Lexicon-Based ABSA** model is highly interpretable and lightweight. Its decision-making process is transparent, as each sentiment prediction can be traced back to specific words and lexicon scores. It is also very fast and easy to debug, making it ideal for smaller datasets or real-time systems. However, its major weakness lies in the lack of contextual awareness, it often misinterprets sarcasm or subtle tone shifts.

- The **Transformer-Based ABSA** approach achieves significantly higher accuracy by leveraging contextual embeddings and deep learning. It understands relationships between aspects and sentiments even when the sentiment is expressed indirectly. This makes it far more robust for complex or multi-aspect sentences. However, it requires more computational power and may run slowly on CPUs. Furthermore, transformers are less interpretable, it is difficult to explain why the model arrived at a certain prediction, which can be problematic in applications requiring transparency.

Discussion

Strengths & weaknesses, challenges faced and possible use cases

- Finally, the **LLM-Based ABSA** model, implemented using a local large language model via Ollama, proved to be the most flexible and context-aware. It can handle irony, mixed sentiments, and even unseen domains without explicit retraining. Its natural language reasoning abilities allow it to generalize far beyond fixed patterns. However, this power comes at the cost of speed, inference is much slower compared to the other methods, and the JSON parsing process sometimes requires additional error handling or regex extraction. Moreover, its resource requirements make it less suitable for large-scale, real-time deployments.

- In summary, the lexicon model excels in simplicity and speed, the transformer model offers the best balance between accuracy and practicality, and the LLM-based model delivers the highest interpretative quality at the cost of efficiency.

Conclusion

This project demonstrated that **Aspect-Based Sentiment Analysis** can be effectively implemented through multiple paradigms:

- **Rule-based (Lexicon)** for interpretability,
- **Pretrained Transformers** for reliability and scalability,
- **LLMs** for contextual reasoning and human-like understanding.

The LLM-based approach achieved the highest qualitative accuracy, particularly for sarcasm, irony, and subtle mixed sentiments.

However, its computational cost makes it less ideal for large-scale deployment compared to the Transformer-based solution.

Conclusion

- **Key learnings:**
  - Unified API design simplifies comparison and testing.
  - Simple datasets can reveal complex model behaviors.
  - LLMs, even locally, are powerful tools for fine-grained NLP when carefully prompted.
- **Future improvements:**
  - Automate aspect extraction via dependency-based templates for Transformers.
  - Extend datasets with multilingual or domain-specific examples.
  - Implement hybrid approaches combining lexicon precision with LLM reasoning.

Thank you for your attention!