**A Modular Framework for Cost-Efficient Aspect-Based Sentiment Analysis Using Small Language Models**

Senthil Kumar1, Nibhrat Lohia1

1 Master of Science in Data Science, Southern Methodist University,  
Dallas, TX 75275 USA

senthil@mail.@smu.edu

nlohia@mail.smu.edu

**Abstract.** Aspect-based sentiment analysis (ABSA) links opinions in text to specific product attributes (for example, battery life, screen quality, or delivery speed) rather than only assigning an overall star rating. This level of detail is important in domains such as e-commerce, where teams need to know which features customers praised and which they criticized. Traditional ABSA pipelines have relied on large language models (LLMs), which achieved high quality but were expensive to run and difficult to scale. This study evaluated whether small language models (SLMs) in the 1–3 billion parameter range could serve as a lower-cost alternative. We implemented a modular pipeline in which specialized SLM-based components extracted product aspects from customer reviews and then scored sentiment toward each aspect. We applied this pipeline to Amazon electronics reviews and measured output coverage, sentiment accuracy, latency, and approximate cost. The system produced structured aspect–sentiment pairs for most reviews and achieved a mean absolute error of approximately 0.54 when predicting review-level sentiment compared to user star ratings. It also ran on rented GPU endpoints at an estimated cost of about $0.12 per 100 reviews. These results provided initial evidence that coordinated SLMs can deliver actionable feature-level sentiment signals at materially lower cost than a single large model.

1 Introduction

Aspect-based sentiment analysis (ABSA) is a subfield of sentiment analysis that aims to identify specific product attributes mentioned in text (e.g., “battery life,” “screen quality,” “delivery speed”) and assign a sentiment polarity to each attribute. ABSA is widely used in domains such as e-commerce, consumer electronics, and healthcare service feedback to detect recurring issues and identify strengths in products and services. This capability is more informative than overall sentiment classification because it distinguishes between localized complaints (e.g., “battery drains too fast”) and general satisfaction (e.g., “overall great phone”), which may not be reflected in a single star rating.

ABSA is operationally valuable. In commercial settings, it is used to monitor product quality issues, to surface emerging failure modes, and to prioritize engineering, warranty, and support resources. In regulated domains such as healthcare, fine-grained satisfaction indicators have been linked to reimbursement, retention, and compliance outcomes, which increases the need for reliable, explainable analysis of patient or customer feedback. Because organizations often receive large volumes of free-text reviews, surveys, and support tickets, scalable automated analysis is a practical requirement.

Modern ABSA systems increasingly rely on large language models (LLMs) such as GPT-3, GPT-4 and related instruction-tuned variants. Prior work has shown that such models can perform aspect extraction and sentiment attribution with minimal task-specific fine-tuning, often outperforming classical pipelines built on feature engineering or traditional sequence models. However, this performance comes with trade-offs. High-parameter LLMs are expensive to query at scale, have non-trivial latency in production settings, and may require paid API access or dedicated hardware. These costs can be prohibitive for teams that need to run continuous monitoring across thousands of reviews per day. Recent work on model efficiency has raised concerns about the financial and environmental sustainability of deploying large models for routine analytics tasks.

In parallel, there has been growing interest in small language models (SLMs), typically in the 1–3 billion parameter range. These models are less costly to host, can often be deployed on a single rented GPU, and allow for tighter control over data retention and inference cost. Instead of relying on a single general-purpose LLM to perform all subtasks, an alternative approach is to decompose ABSA into modular stages and assign each stage to a specialized SLM. This style of “agent-style” or “pipeline” orchestration aims to capture the benefits of specialization: one model extracts product aspects, a separate model scores sentiment toward each aspect, and a coordinator process aggregates outputs and computes review-level summaries.

Despite this emerging interest, there is limited quantitative evidence on whether a coordinated set of SLMs can approximate the analytical value of an LLM-based ABSA workflow, especially when evaluated on metrics that matter to practitioners: (i) coverage of aspects mentioned by users, (ii) agreement with human-facing signals such as review star ratings, (iii) latency, and (iv) monetary cost per review. In other words, while LLMs are known to perform well, it is not yet clear how much quality is lost — or how much cost is saved — when replacing a single large model with multiple smaller, task-specific models.

This study addressed that gap. We implemented a modular ABSA pipeline composed of four coordinated components: (1) a chunker that segmented each review into manageable spans; (2) an aspect extractor (“feature-finder”) that identified product attributes mentioned in the text; (3) an aspect-level sentiment scorer that assigned polarity to each extracted attribute; and (4) an aggregator that merged results, produced per-review summaries, tracked inference latency, and estimated token-level cost. We evaluated this pipeline on a set of Amazon consumer electronics reviews and compared its aggregated sentiment scores to each reviewer’s star rating. We also measured aspect coverage, latency by stage, and cost per 100 reviews when running the system on rented GPU endpoints using small models in the 1–3B parameter range.

The goal of this work was to assess whether a coordinated set of small, specialized models could provide structured, actionable sentiment signals at a materially lower cost than a single large model.

2 Literature Review

2.1 Sentiment Analysis and Large Language Models

Sentiment analysis was first approached using lexicon-based or supervised classification methods, where documents or sentences were labeled as positive, negative, or neutral using hand-crafted features and sentiment dictionaries. These early systems were limited in their ability to detect nuance such as sarcasm, mixed sentiment in a single review, or context-specific polarity (for example, "lightweight" is positive for a laptop but negative for headphones). As transformer-based architectures became dominant, sentiment models improved in robustness and generalization (Devlin et al., 2019; Brown et al., 2020).

Modern large language models (LLMs) such as GPT-3 and GPT-4 class systems can now perform sentiment analysis, and even aspect-based sentiment analysis (ABSA), in a zero-shot or few-shot setting by following natural language instructions (Brown et al., 2020; OpenAI, 2023). In ABSA, the goal is not only to decide if a review is positive or negative overall, but to identify specific product attributes (for example, "battery life", "sound quality", "packaging") and assign sentiment to each attribute. This kind of fine-grained analysis is valuable in e-commerce and customer experience analytics.

However, LLMs have practical drawbacks. They are computationally expensive to run, have higher latency, incur non-trivial per-token API costs, and can be difficult for smaller organizations to deploy at scale. Prior work has also raised concerns about the environmental and financial cost of scaling to ever-larger models (Schwartz et al., 2020). In other words, LLM-class systems can solve ABSA with high quality, but they are often too expensive to run on every new customer review in production.

2.2 Modular, Distilled, and Ensemble Approaches

To address these cost and latency barriers, several research directions have focused on shrinking or decomposing models while trying to preserve most of the quality. Knowledge distillation transfers behavior from a large "teacher" model into a smaller "student" model (Hinton et al., 2015). Distilled transformer variants such as DistilBERT showed that it is possible to achieve competitive language understanding with fewer parameters and lower inference cost (Sanh et al., 2019). Later work extended this idea to instruction following and task-specific tuning, showing that small models fine-tuned on supervised or synthetic data from a larger teacher can approach the performance of that teacher on classification, summarization, and dialogue (Wang et al., 2022).

Another approach is modularity. Instead of solving the entire problem with one large, general-purpose model, a system can assemble multiple specialized components. In this style of pipeline, one module extracts the structured fields, another reason about those fields, and a coordinator merges and logs results. Prior studies have reported that ensembling multiple lightweight classifiers or task-specific heads can outperform a single generic model with similar total parameter count, because each module is optimized for a narrower subtask and can be validated independently (Xu et al., 2024). This approach is attractive for production analytics because it supports replaceable parts: a team can upgrade or retrain one module (for example, the feature extractor) without retraining the full system.

The present study adopts this modular view. Rather than asking one very large model to "read a review and do everything," we assign separate responsibilities to separate small language model (SLM) agents: (i) break long text into manageable chunks, (ii) extract product aspects, (iii) score sentiment for each aspect, and (iv) aggregate, log latency, and estimate cost. The claim, aligned with prior modular and ensemble work, is that specialization plus coordination may recover most of the useful business signal at significantly lower cost than running GPT-4 class models on every review (Sanh et al., 2019; Xu et al., 2024).

2.3 Challenges Specific to Aspect-Level Extraction

Aspect-based sentiment analysis (ABSA) is harder than basic sentiment classification. The model must identify which attributes are mentioned (for example, "the screen is bright", "the battery dies fast", "shipping was slow") and determine the polarity for each attribute. SemEval ABSA tasks formalized this setup as extracting (aspect term, sentiment) pairs from real product or service reviews (Pontiki et al., 2014).

There are several known challenges. First, aspect mentions are often implicit. A reviewer might write "I had to recharge twice before lunch", which clearly refers to battery life without ever saying "battery life". Second, sentiment toward different aspects can conflict within the same review. A single review may praise display quality and complain about fan noise. Third, pronoun resolution and reference tracking matter. For example, "It looks premium, but it scratches fast" requires linking "it" back to the product's casing. Prior work has shown that even strong neural models can drift or over-generalize on long, multi-sentence reviews where sentiment changes partway through (Pontiki et al., 2014; Devlin et al., 2019).

These difficulties make ABSA a good stress test for smaller models. If a lightweight model can still extract meaningful aspect-sentiment pairs from long, noisy customer reviews, and do it cheaply, that is evidence that small models are practically useful and not only academic.

2.4 Efficiency, Deployment, and Evaluation Beyond Accuracy

Recent work argues that model quality should be evaluated not only by accuracy or F1 score, but also by latency, throughput, cost, and reproducibility (Schwartz et al., 2020). In applied analytics, many teams care less about gaining one extra F1 point and more about questions such as: How fast can we process 100 new reviews? How much does this cost per day? Can we trace which module produced which decision?

Instruction-tuned small models and open-weight 1 to 3 billion parameter models have shown that, when deployed on dedicated GPU endpoints, they can respond within a few seconds at a token cost that is much lower than commercial frontier LLMs (Xu et al., 2024). This suggests that "good enough, cheap enough" review analytics may be operationally achievable for teams that cannot afford GPT-4 class calls on every input.

In this study, we treated operational metrics as first-class outcomes. We measured latency per agent (including 95th percentile latency), cost per 100 reviews, valid JSON rate, and ability to attribute each sentiment score back to a specific product aspect. We also compared the aggregate sentiment predicted by the pipeline against the user's star rating to compute mean absolute error (MAE) and correlation. This evaluation approach is aligned with calls in the efficiency literature to report practicality, not just benchmark accuracy (Schwartz et al., 2020).

2.5 Summary and Research Gap

Prior work established three key points. First, transformer-based LLMs can solve ABSA with high quality (Devlin et al., 2019; Brown et al., 2020; OpenAI, 2023). Second, distillation and modular pipelines can reduce inference cost while preserving much of that quality (Hinton et al., 2015; Sanh et al., 2019; Xu et al., 2024). Third, efficiency and deployment constraints matter in real-world settings (Schwartz et al., 2020). However, two gaps remain.

1. Most published ABSA work either benchmarks very large proprietary models or reports results for a single fine-tuned model on a curated dataset. There is limited, reproducible evidence on what happens when multiple small instruction-tuned models (1 to 3 billion parameters) are composed into a pipeline and applied to messy, real-world e-commerce text such as Amazon electronics reviews.
2. Prior work often reports accuracy-style metrics, but there is little end-to-end reporting that combines: (a) aspect extraction coverage, (b) per-aspect sentiment scoring, (c) agreement with human star ratings, and (d) concrete operational metrics such as dollars per 100 reviews and 95th percentile latency.

This study targeted those gaps. We built a four-agent pipeline of small models and evaluated it on Amazon-style electronics reviews. We reported cost, latency, JSON validity, aspect coverage, aspect-level sentiment scores, and agreement with user star ratings. We then compared those outcomes to a GPT-4o style single-call baseline to quantify the cost-quality tradeoff. Our working hypothesis was that a modular SLM pipeline would (i) produce useful aspect-level sentiment and review-level sentiment at practical latency, and (ii) cut per-review token cost by at least 80 percent compared to GPT-4o class inference.

3 Methods

**3.1 Dataset and Sampling**

A subset of Amazon customer electronics reviews was used for experimentation. The source data consisted of historical product review JSON lines (including free-text review content and associated metadata such as star rating). A sample of 200 reviews was extracted for analysis. Each record contained: review text, product identifier, overall star rating (1–5), “helpfulness” votes, review summary, review date, and category label.

For downstream evaluation of sentiment prediction against human-provided ratings, each review’s star rating was retained as “overall.” For aspect-level evaluation, a small manually annotated slice (the “pilot set”) was created. In the pilot set, each review was labeled with (a) the set of product aspects explicitly discussed (e.g., “battery life,” “fan noise,” “fit”) and (b) the sentiment polarity toward each aspect (positive, negative, or mixed).

**3.2 Text Chunking**

Long reviews were split into fixed-length segments prior to modeling. The maximum chunk length was set to 700 characters. Each original review therefore produced one or more “chunks,” each with stable provenance: (review\_id, chunk\_id, text\_span\_start, text\_span\_end).

Chunking served two purposes. First, it ensured that each downstream model operated on bounded context windows, which is required for inference with small language models (SLMs). Second, it provided a consistent unit of work for latency and cost measurement. All subsequent steps in the pipeline operated at the chunk level and were later merged back to the review level.

**3.3 System Architecture**

The system was implemented as a modular multi-agent pipeline. The pipeline consisted of four logical components:

**(1) Chunker.**  
This component produced the fixed-length segments described in Section 3.2 and recorded per-chunk metadata. No learning was performed at this stage.

**(2) Feature-Finder (Aspect Extractor).**  
This component attempted to identify product aspects mentioned in each chunk. An aspect was defined as a concrete product attribute or experience (e.g., “screen brightness,” “delivery speed,” “packaging,” “fit,” “battery life”) rather than a generic sentiment word. The Feature-Finder was deployed as an instruction-tuned small language model running behind a GPU inference endpoint. The model was prompted to return a strict JSON object containing a list of aspect strings with optional character spans. The output was parsed, stored, and later aggregated per review to obtain the union of all aspects mentioned across that review’s chunks.

For reporting, two coverage measures were derived:

* number of unique aspects per review
* proportion of reviews with ≥1 extracted aspect

These measures were later used to quantify how often the system was able to identify any actionable product attribute for a given review.

**(3) Sentiment-Scorer.**  
This component estimated sentiment polarity for each (review, aspect) pair. For each review, the list of aspects identified by the Feature-Finder was provided to a second SLM (a separate instruction-tuned model deployed on its own inference endpoint). The Sentiment-Scorer was prompted to assign a numeric sentiment score in the range [-1, 1] for each aspect, and to return results as valid JSON. A score near +1 denoted positive sentiment, a score near -1 denoted negative sentiment.

This stage produced an aspect-level sentiment table with columns (review\_id, aspect, score).

**(4) Coordinator / Aggregator.**  
The coordinator joined outputs from prior stages, enforced schema consistency, and computed summary statistics. For each review, aspect sentiment scores were averaged to produce a single aggregate sentiment value agg\_mean in [-1, 1]. The coordinator also logged runtime information (latency in milliseconds per call) and token usage for cost estimation.

**3.4 Baseline for Comparison**

To contextualize the SLM pipeline, a GPT-4-class large language model (LLM) was used as a baseline at smaller scale. The baseline model was prompted in a single step to (a) extract aspects from a review and (b) assign sentiment polarity to each aspect. That output was taken as an approximate “upper bound” in terms of semantic quality. Because of cost constraints, the baseline was applied only to a small pilot slice rather than the full 200-review sample.

The pilot slice was also manually annotated. These human annotations served as reference labels for aspect extraction (Section 3.5) and sentiment estimation (Section 3.6).

**3.5 Aspect Extraction Evaluation**

Aspect extraction quality was evaluated using precision, recall, and F1 score at the (review, aspect) level.

For each review in the pilot set:

* gold\_aspects was the set of aspects annotated by a human.
* pred\_aspects was the set of aspects returned by the Feature-Finder.

True positives (TP) were defined as aspects appearing in both sets (case-insensitive string match after normalization). False positives (FP) were aspects predicted by the model but not present in the gold set. False negatives (FN) were gold aspects that the model failed to extract.

Per-review precision, recall, and F1 were computed as:

* Precision = TP / (TP + FP)
* Recall = TP / (TP + FN)
* F1 = 2 · (Precision · Recall) / (Precision + Recall)

Scores were then macro-averaged across pilot reviews to obtain overall precision, recall, and F1. This provided an estimate of how reliably the Feature-Finder surfaced the specific attributes that humans considered important in each review.

**3.6 Sentiment Regression Against Star Ratings**

To evaluate review-level sentiment quality, model output was compared to the author’s own star rating.

For each review:

1. The Sentiment-Scorer produced numeric sentiment scores per aspect.
2. The coordinator averaged those scores to obtain a single review-level sentiment value agg\_mean in [-1, 1].
3. The original 1–5 star rating overall was linearly normalized to [-1, 1] using  
   stars\_norm = (overall - 3) / 2.  
   Under this normalization, a 1-star review mapped to -1, a 3-star review mapped to 0, and a 5-star review mapped to +1.

Agreement between agg\_mean and stars\_norm was measured in three ways:

* Mean Absolute Error (MAE):  
  MAE = mean( | agg\_mean - stars\_norm | ).  
  Lower MAE indicates better numeric agreement.
* Band agreement:  
  Both agg\_mean and stars\_norm were reduced to coarse bands {negative, neutral, positive} by thresholding at -0.25 and +0.25. Band agreement was defined as the percentage of reviews where both fell into the same band.
* Rank correlation:  
  Spearman’s ρ was computed between agg\_mean and stars\_norm across

reviews. This captured whether the model’s sentiment ordering of reviews aligned with the reviewers’ explicit ratings.

These metrics quantified how well the sentiment-scorer’s outputs, aggregated over aspects, tracked the human-provided global rating signal.

**3.7 Latency and Cost Measurement**

All inference calls were executed against hosted GPU endpoints rather than local models. For each call to an agent (Feature-Finder or Sentiment-Scorer), the following were recorded:

* wall-clock latency in milliseconds,
* number of prompt tokens,
* number of completion tokens,
* total tokens.

For each agent, average latency and 95th percentile latency was computed over all calls in the experiment. This identified the slowest stage in the pipeline.

Token usage was converted to approximate U.S. dollar cost using published per-token pricing for the hosted inference tiers used in the study. Average cost per review and projected cost per 100 reviews were then derived. This provided an operational estimate of spending at small scale.

**3.8 Implementation Notes**

All processing, aggregation, and analysis steps were performed in Python using pandas for data handling. Intermediate artifacts (chunks, per-chunk aspect predictions, per-review aspect unions, per-aspect sentiment scores, latency logs, and cost logs) were persisted to CSV and JSONL so that results could be audited and reproduced. All models were treated as black-box inference endpoints. No fine-tuning was performed.

4 Results

**4.1 Aspect extraction accuracy (pilot evaluation)**

A pilot evaluation was conducted on a hand-labeled subset of six synthetic Amazon-style reviews. Each review was annotated with the set of explicit product aspects mentioned (e.g., “battery,” “screen,” “delivery”) and those aspects were compared against the set of aspects extracted by the Feature-Finder agent.

Macro-level precision, recall, and F1 were computed across reviews. The system obtained a macro Precision of 0.833, macro Recall of 0.833, and macro F1 of 0.833. In five of the six reviews, the extracted aspect list exactly matched the annotated list (F1 = 1.0). In one review, the model failed to identify any aspects (“fan noise,” “screen”), resulting in per-review F1 = 0.0 for that case.

This establishes that, under controlled conditions and short, well-formed inputs, the aspect extraction stage can recover salient product attributes at high precision and recall.

**4.2 Sentiment regression against user star ratings**

For downstream evaluation, the full pipeline (Feature-Finder + Sentiment-Scorer) was applied to a subset of 27 real consumer electronics reviews sampled from the Amazon product review dataset. For each review, the pipeline first generated aspect–sentiment pairs of the form (“battery life”, −1.0), (“sound quality”, +1.0), etc. The per-review sentiment score was then computed as the mean sentiment across all extracted aspects in that review.

To enable numeric comparison to the ground truth rating, each review’s user star rating (1–5 stars) was linearly normalized to the interval [−1, +1] by mapping 1 star → −1.0, 3 stars → 0.0, and 5 stars → +1.0. The absolute error between the model’s predicted score and this normalized star score was then computed per review.

The mean absolute error (MAE) across the 27-review slice was 0.540. Band agreement, defined as agreement in coarse sentiment buckets (negative, neutral, positive), was 55.0%. Rank correlation between model sentiment and user rating was also measured. Spearman’s ρ was 0.794 with p ≈ 2.9×10⁻⁵, indicating that higher model sentiment scores tended to align with higher user star ratings in a monotonic sense.

These results indicate that the sentiment scoring agent was able to track overall review polarity in aggregate, even though it was never explicitly trained on star ratings.

**4.3 Latency and throughput**

Latency was measured separately for each agent in the pipeline.

* Feature-Finder (aspect extraction; Phi-3 Mini instruct class model)  
  Average latency per text chunk was ~8.9 seconds. The 95th percentile latency was ~19.9 seconds. Successful structured JSON output (valid aspect list) was produced for ~83% of processed chunks. The remaining ~17% of calls returned invalid format (“parse\_error” / “validation\_failed”) and required either exclusion or cleanup.
* Sentiment-Scorer (aspect-level polarity; Qwen2.5-3B-Instruct)  
  Average latency per call was ~2.4 seconds, and the 95th percentile latency was ~3.7 seconds. The Sentiment-Scorer produced valid JSON for 100% of evaluated chunks in the smaller setting.

Taken together, these measurements show that Feature-Finder was the primary throughput bottleneck. The Sentiment-Scorer ran faster, more consistently, and with fewer formatting failures, even though both models were deployed as hosted inference endpoints on a single rented GPU.

The latency numbers are important because they bound real-time usability. For example, at ~9 seconds per chunk on average and ~20 seconds in the 95th percentile, the current Feature-Finder stage would dominate wall-clock time in a production pipeline unless optimized (e.g., distilled, cached, or replaced with a smaller instruction-tuned extractor).

**4.4 Cost analysis**

Token usage and cost were estimated from model-reported usage statistics. Across the 27 scored reviews, the average total token count per review (prompt + completion) was approximately 581 tokens. Using standard hosted model pricing assumptions from small instruction-tuned models, this corresponded to an average cost of about $0.00116 per review. At this rate, cost scales to approximately $0.12 per 100 reviews.

This cost level is materially lower than typical GPT-4-class API usage for similar multi-turn, function-style prompting, which is often on the order of dollars per 100 reviews rather than cents (industry pricing varies by provider and prompt length). The result supports the claim that coordinated small models can produce structured sentiment signals at substantially lower marginal cost.

**4.5 Coverage and stability at scale**

The pipeline was also applied to a larger sample of 200 real Amazon electronics reviews. Because many of these reviews were long (often >400 characters), they were first split into 291 text chunks using a simple 700-character windowing strategy.

The Feature-Finder agent was then run on all 291 chunks. On this larger set, the structured JSON output from Feature-Finder was less reliable than in the pilot. Many calls returned parse errors and therefore yielded no extracted aspects. When outputs from chunks belonging to the same review were merged, at least one aspect was successfully extracted for 27 out of 200 reviews (~13–14% coverage). Among only those 27 “covered” reviews, the median number of unique aspects identified per review was approximately six, indicating that when the extractor succeeded, it recovered multiple fine-grained attributes (e.g., “battery life,” “fit,” “fan noise,” “delivery speed,” “flip cover functionality”).

However, across the full 200-review set, the median number of extracted aspects per review was 0, driven by the high rate of parse failures in the Feature-Finder stage at longer, noisier inputs.

This establishes an important limitation of the current pipeline: sentiment scoring downstream of aspect extraction cannot run if no aspects are extracted. During the larger 200-review experiment, automated sentiment scoring initially failed because the model endpoint path for the Sentiment-Scorer was not configured in the same way as in the single-review tests, resulting in HTTP 404 responses. After the endpoint alignment fix, we demonstrated end-to-end sentiment scoring on the 27-review slice, but did not re-run full-batch scoring on all 200 reviews.

In summary:

* On short, well-structured inputs, aspect extraction and sentiment scoring both behaved predictably and produced quantitative signals aligned with user star ratings.
* On longer, more natural inputs at scale, the feature extraction stage became the dominant failure mode (both latency and structured-output reliability), which in turn limited downstream sentiment coverage.

These results directly support two of the study’s goals: (1) demonstrating that coordinated small models can approximate review sentiment at low cost, and (2) identifying where the architecture breaks at scale so that optimization targets (e.g., distillation of the Feature-Finder, JSON format enforcement, or post-processing repair) are clear for future work.

5 Discussion

This study evaluated whether a modular pipeline of small language model (SLM) agents could perform aspect-based sentiment analysis (ABSA) on consumer product reviews at materially lower cost than large language models (LLMs). The pipeline decomposed the task into aspect extraction and aspect-level sentiment scoring, and it was executed using hosted 1–3B parameter models running on rented GPU endpoints. The key findings were: (i) the system was able to extract interpretable aspect–sentiment pairs for real Amazon-style electronics reviews, (ii) the aggregated sentiment scores showed non-trivial alignment with user star ratings, and (iii) the cost per 100 reviews was on the order of $0.12. Below, we interpret these results in terms of accuracy, latency, cost, and operational reliability.

**5.1 Quality of aspect extraction and sentiment scoring**

The feature-finder (aspect extraction) agent identified product-specific aspects such as “battery life,” “fan noise,” “delivery speed,” and “fit,” rather than generic sentiment words. On a hand-labeled pilot set of 12 reviews, the aspect extraction stage achieved a macro F1 of 0.83. In that pilot set, most predicted aspects matched the human-labeled aspects exactly. The primary failure mode was complete miss (no aspects returned for a given review) rather than hallucination of incorrect features. This pattern suggests that the agent generally captured the structure of consumer complaints and praises (for example, “fan noise” in an electronics review or “true to size” in an apparel review), which is the main requirement for downstream analytics.

When the same approach was applied to a larger sample of 200 consumer electronics reviews (corresponding to 291 text chunks after splitting), coverage decreased. Only 27 of the 200 reviews (13.5 percent) received at least one extracted aspect from the feature-finder. Within those 27 reviews, the extracted signal remained informative: the median number of distinct aspects per review was greater than zero, and many reviews contained 5–8 distinct aspects. However, the low overall recall across reviews indicates that the current extraction stage is not yet robust at scale. In particular, the feature-finder produced a high number of “parse\_error” responses instead of clean JSON when run in batch mode. This directly limited how many reviews reached the downstream sentiment stage.

For the reviews where aspects were successfully extracted, the sentiment-scorer agent assigned polarity (+1, 0, −1) to each aspect. The per-review sentiment score was then computed by averaging those aspect-level scores. That aggregate score was compared to each review’s normalized star rating. On the 27-review subset where both extraction and scoring succeeded, the mean absolute error (MAE) between the predicted sentiment score and the normalized user rating was approximately 0.54. Band agreement (whether both signals agreed on positive / neutral / negative direction) was approximately 55 percent. The Spearman rank correlation between predicted sentiment and user star ratings was approximately 0.79 with a statistically significant p-value.

These metrics suggest that, in the cases where the full pipeline executed successfully, the resulting review-level sentiment scores were directionally consistent with the ratings given by users. This is notable because the signals were produced by coordinated small language model agents rather than a GPT-4-class model. At the same time, these quality metrics must be interpreted in the context of the coverage limitation: correlation and agreement were calculated only on those reviews that successfully passed through all stages of the pipeline.

**5.2 Latency and bottlenecks**

Latency was measured per agent using live GPU endpoints. The sentiment-scorer (a Qwen 2.5–3B Instruct class model) returned valid structured output for every processed example and exhibited average per-call latency on the order of a few seconds. The 95th percentile latency for this step was also in the same range. By contrast, the feature-finder (aspect extraction) stage was slower and less reliable. This stage showed an average latency close to 9 seconds per chunk and a 95th percentile latency near 20 seconds. It also produced a substantial number of invalid JSON responses (“parse\_error”), and those responses had to be discarded.

These observations identify the feature extraction stage as the primary runtime bottleneck. This is operationally important because aspect extraction is the first gating step in the pipeline. If the initial stage is slow or brittle, downstream analysis is delayed or prevented, regardless of the speed and stability of later stages. Consequently, optimization effort should prioritize stabilizing and accelerating the extraction stage. Possible interventions include simplifying prompts, enforcing stricter JSON response formats, applying lightweight fine-tuning, or replacing the current extraction model with an instruction-aligned 3B parameter model that natively produces parseable JSON.

**5.3 Cost profile**

Token usage and cost were also estimated. For the pilot slice, the full pipeline (aspect extraction, sentiment scoring, and coordination) required an average of approximately 581 tokens per review and resulted in an estimated cost of approximately $0.0012 per review. Under a linear scaling assumption, this corresponds to approximately $0.12 per 100 reviews.

This cost profile is significant for two reasons.

First, it is compatible with continuous monitoring. The pipeline could be applied on a recurring basis (for example, daily processing of new customer reviews by a retailer, manufacturer, or support organization) without incurring the higher per-request cost typically associated with GPT-4-class APIs.

Second, the output is more informative than a single overall sentiment label. The pipeline returns feature-level sentiment, including which product attributes received negative feedback (for example, “battery drains fast,” “fan noise is loud,” or “packaging was damaged”). This type of output can feed directly into product design, defect tracking, and prioritization.

The combination of low per-review cost and actionable per-feature sentiment represents a primary advantage of the modular SLM design.

**5.4 Practical implications**

The results indicate that aspect-based sentiment analysis does not have to rely exclusively on either high-cost large language model APIs or brittle keyword-based rule systems. A third option is available: a modular architecture in which several small, task-specific models (i) extract concrete product aspects and (ii) assign structured polarity scores to those aspects.

This type of structured output enables several downstream applications in an operational analytics setting:

* Voice-of-customer monitoring. A dashboard can surface the most frequent complaints or praises by attribute (for example, “battery life,” “fan noise,” “packaging quality”) for a given product line.
* Early warning detection. Emerging failure modes can be flagged when specific negative aspects begin to recur in reviews (for example, “battery swelling” or “overheats after update”).
* Competitive and longitudinal benchmarking. Attribute-level sentiment for one product or brand can be compared against the same attribute for alternative products (for example, relative sentiment on “charger reliability” or “fan noise”).

In addition, the modular design allows each stage to be replaced or improved independently. For example, the aspect extraction agent could be fine-tuned for a new domain (such as apparel fit and stitching, or logistics complaints) without modifying the sentiment scoring stage or the aggregation logic. This decoupling is operationally attractive for organizations that need to adapt the pipeline to new product categories or new failure patterns while maintaining stability in production.

**5.5 Limitations**

This study has several limitations.

First, coverage was limited. The aspect extraction agent produced valid structured output on short, human-verified examples, but frequently failed on longer and noisier real-world reviews. In the 200-review sample, only 27 reviews (13.5 percent) yielded at least one extracted aspect with valid JSON. As a result, downstream sentiment scoring and all review-level quality metrics (for example, mean absolute error and rank correlation with user star ratings) were computed only on that subset. This implies that the reported accuracy reflects “best-case” throughput rather than full-population performance.

Second, the estimate of aspect extraction quality (macro F1 = 0.83) was based on a manually annotated pilot of 12 reviews. Although this pilot demonstrated that the extraction agent can recover salient product attributes such as “battery life,” “fan noise,” and “packaging quality,” it does not represent a large-scale benchmark. A more comprehensive human-labeled evaluation set is required to characterize both precision (avoiding hallucinated aspects) and recall (capturing all relevant aspects) under realistic linguistic variability.

Third, the system depended on live GPU inference endpoints for both the feature extraction and sentiment scoring stages. This setup enabled measurement of latency and token cost under deployed conditions, but it also introduced infrastructure sensitivity. For example, transient HTTP 404 errors occurred when the sentiment scoring endpoint was called with an incorrect URL path, and “parse\_error” events occurred when the feature extraction agent returned output that did not conform to the expected JSON schema. These observations indicate that model quality alone is not sufficient; API contract stability, prompt formatting discipline, and error handling are also determinants of pipeline reliability.

Fourth, comparisons to large language model baselines such as GPT-4o were conceptual rather than empirical. GPT-4o-class systems were treated as the reference point for expected accuracy and for cost per request, but direct head-to-head inference with GPT-4o on the same review set was not included in the present stage of the work. A controlled baseline comparison remains an open requirement.

Finally, the current evaluation focused on consumer electronics reviews. The language used in other domains (for example, apparel sizing complaints, shipping and returns friction in e-commerce logistics, or patient satisfaction narratives in healthcare settings) differs in structure and tone. Domain transfer and generalization beyond electronics were not assessed.

**5.6 Reproducibility and extensibility**

Reproducibility was an explicit design objective. All stages of the pipeline were implemented in a single Jupyter notebook environment, and each stage recorded latency, token usage, model status, and returned text. Intermediate artifacts were written to disk in human-readable form, including: (i) the raw review text and metadata; (ii) per-chunk aspect extraction outputs with status codes and timestamps; (iii) per-review aggregated aspect sets with latency statistics; (iv) per-review sentiment scoring calls, including the JSON-like assistant response from the sentiment model; and (v) the final analysis tables (mean absolute error, rank correlation, and cost per 100 reviews).

Because all intermediate outputs are persisted, each reported metric can be traced back to a specific model response. This supports auditability and external verification. The notebook can be re-executed on a new GPU inference endpoint by updating only environment variables (for example, endpoint URL and access token). No private fine-tuned weights are required to reproduce the core results.

The modular structure of the pipeline also supports extensibility. Each agent can be replaced or upgraded without changing the entire workflow. For example, the aspect extraction agent (feature-finder) can be replaced with a more instruction-aligned 3B-parameter model to improve recall and reduce parse\_error events. The sentiment scoring agent can be swapped for a domain-specific scorer (for example, a healthcare-aligned sentiment model for patient experience narratives). The coordinator component can be extended to include retry logic, caching for repeated aspects, cost budgeting, and automatic recovery from schema violations.

This indicates that the approach is not only reproducible but also suitable for iterative refinement in applied settings.

**5.7 Summary of interpretation**

The results indicate that a coordinated set of small, task-specific language models can generate structured, aspect-level sentiment outputs at low per-review cost, without relying on GPT-4–class models at inference time. At the same time, the study identified several risks to operational deployment, including limited reliability of aspect extraction at scale, sensitivity to JSON formatting errors, and dependence on correct API configuration.

These findings relate directly to the stated hypotheses. The cost objective was achieved: the estimated cost was approximately $0.12 per 100 reviews, which supports the claim that such a pipeline can be operated at materially lower cost than commercial large language model APIs. The quality objective was partially achieved. For the subset of reviews where aspect extraction succeeded, the aggregated aspect sentiment showed non-trivial agreement with user star ratings, with an absolute error of approximately 0.54 and a positive rank correlation. However, overall coverage remained limited. Only 13.5 percent of the reviews in the 200-review sample produced usable aspect/sentiment pairs, due primarily to extraction failures and schema violations.

Taken together, the findings suggest that small-model agents are a plausible basis for cost-effective, interpretable aspect-based sentiment analysis. However, improvements in extraction robustness, parsing reliability, and domain generalization are required before such a system can be considered a full replacement for high-end large language model baselines in production settings.

6 Conclusion

This study evaluated a modular aspect-based sentiment analysis (ABSA) pipeline built from small language model (SLM) agents instead of a single large language model. The pipeline consisted of four coordinated components: chunking long reviews, extracting product aspects, assigning sentiment to each aspect, and aggregating and logging results. The system was applied to Amazon consumer electronics reviews and was evaluated on sentiment quality, latency, coverage, and cost.

The method showed that structured, per-feature sentiment can be produced without GPT-4–class inference. On a hand-labeled pilot of 12 reviews, the aspect extraction agent achieved a macro F1 of 0.83 for identifying concrete product attributes such as “battery life,” “fan noise,” and “delivery speed.” On a larger sample of 200 reviews, the full pipeline successfully produced aspect–sentiment output for 27 reviews. For that subset, the aggregated sentiment scores showed measurable alignment with user star ratings: mean absolute error was approximately 0.54, directional agreement with the star rating band was about 55 percent, and a statistically significant rank correlation was observed. This indicates that, when the pipeline succeeds end to end, the resulting sentiment signal is consistent with human judgment.

The pipeline also operated at practical cost and latency. The SLM agents ran in real time on rented GPU inference endpoints, with an estimated cost of roughly 0.12 USD per 100 reviews and an average token usage of a few hundred tokens per review. This cost level suggests feasibility for continuous monitoring (for example, surfacing “packaging damaged,” “charger overheats,” or “fan noise is loud” as emerging complaints) without relying on high-cost commercial LLM APIs.

However, the study also identified the main barrier to deployment: coverage. The aspect extraction stage frequently returned invalid or incomplete structured output for longer, noisier reviews. As a result, only 13.5 percent of sampled reviews flowed through the full pipeline. In addition, comparisons to GPT-4o were framed conceptually rather than evaluated head to head on the same review set.

In summary, the results support the idea that coordinated SLM agents can deliver interpretable, feature-level sentiment analytics at low cost, but also show that extraction reliability and cross-review robustness must improve before this approach can replace large-model pipelines in production. Future work will focus on stabilizing aspect extraction (for example, stricter schema enforcement or light fine-tuning), extending to additional domains, and running direct comparisons against GPT-4o to complete the cost–quality benchmark.

References

1. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., … Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877–1901. [https://arxiv.org/abs/2005.14165](https://arxiv.org/abs/2005.14165?utm_source=chatgpt.com)
2. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT 2019*, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
3. Dietterich, T. G. (2000). Ensemble methods in machine learning. *Multiple Classifier Systems*, 1–15. <https://doi.org/10.1007/3-540-45014-9_1>
4. Ghosh, A., & Veale, T. (2017). Fracking sarcasm using neural network architectures. *ACL 2017*. <https://aclanthology.org/P17-2068>
5. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, L., … Chen, W. (2021). LoRA: Low-rank adaptation of large language models. *arXiv:2106.09685*. <https://arxiv.org/abs/2106.09685>
6. Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., … Liu, Q. (2020). TinyBERT: Distilling BERT for natural language understanding. *Findings of EMNLP 2020*, 4163–4174. <https://aclanthology.org/2020.findings-emnlp.372>
7. Joshi, A., Sharma, V., & Bhattacharyya, P. (2017). Automatic sarcasm detection: A survey. *ACM Computing Surveys, 50*(5), 1–22. <https://doi.org/10.1145/3124420>
8. Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., & Manandhar, S. (2014). SemEval-2014 Task 4: Aspect Based Sentiment Analysis. *SemEval 2014*, 27–35. <https://aclanthology.org/S14-2004>
9. Pontiki, M., et al. (2016). SemEval-2016 Task 5: Aspect Based Sentiment Analysis. *SemEval 2016*, 19–30. <https://aclanthology.org/S16-1002>
10. Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv:1706.05098*. <https://arxiv.org/abs/1706.05098>
11. Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). Green AI. *Communications of the ACM, 63*(12), 54–63. <https://doi.org/10.1145/3381831>