Scientific Annals of Economics and Business
73 (SI), 2026, 147-168
DOI: 10.47743/saeb-2026-0027

Listening at Scale: A Replicable, Low-Cost Pipeline for Collecting and Analyzing Google Business Reviews with SerpApi and VADER

Marian Pompiliu Cristescu[*], **Dumitru Alexandru Mara**[**], **Ana-Maria Constantinescu**[***], **Ioana Petrea**[§], **Ana Englitera Visan**[°]

Abstract: Small and medium-sized enterprises increasingly rely on public review platforms as signals of quality, reputation, and customer experience, yet many owners still lack a practical way to process multilingual review text at regular intervals. This study develops a low-cost and reproducible workflow for collecting Google Business reviews, filtering translated English content, and summarizing sentiment with open analytical tools. The pipeline uses SerpApi to retrieve recent reviews for a known Google Maps place identifier and VADER to classify translated review text into positive, neutral, and negative sentiment. The procedure is demonstrated on 200 Google reviews for an auto-rental location, Klass Wagen Sibiu. Results indicate a strongly positive corpus, with 85% positive, 4% neutral, and 11% negative reviews, a mean VADER compound score of 0.538, and a median of 0.733. The association between VADER scores and star ratings is high, r = 0.83, but it is interpreted as convergent descriptive evidence rather than classification accuracy. The study contributes an applied decision-support blueprint that connects online-review analytics, sentiment analysis, and SME monitoring practice. Its limits remain important: review data are self-selected, platform access is constrained, automatic translation can alter sentiment, and lexicon-based models may miss sarcasm or domain-specific wording.

Keywords: sentiment analysis; VADER; SerpApi; Google Maps review; SMEs.

JEL classification: C88; C55; L86.

[*] Lucian Blaga University of Sibiu, Romania; e-mail: *marian.cristescu@ulbsibiu.ro*.
[**] Lucian Blaga University of Sibiu, Romania; e-mail: *dumitrualexandru.mara@ulbsibiu.ro*.
[***] Lucian Blaga University of Sibiu, Romania; e-mail: *anamaria.constantinescu@ulbsibiu.ro* (corresponding author).
[§] Independent Researcher, Sibiu, Romania; e-mail: *ioana.petrea@ulbsibiu.ro*.
[°] Independent Researcher, Bucharest, Romania; e-mail: *englitera.visan@gmail.com*.

Article history: Received 16 March 2026 | Accepted 3 July 2026 | Published online 12 August 2026

To cite this article: Cristescu, M. P., Mara, D. A., Constantinescu, A.-M., Petrea, I., Visan, A. E. (2026). Listening at Scale: A Replicable, Low-Cost Pipeline for Collecting and Analyzing Google Business Reviews with SerpApi and VADER. *Scientific Annals of Economics and Business*, 73(SI), 147-168. https://doi.org/10.47743/saeb-2026-0027.

1. INTRODUCTION

Public review platforms, including Google Maps, Yelp, TripAdvisor, and Trustpilot, have become part of the market infrastructure through which local businesses are discovered, compared, and judged. Prior research shows that online reviews can influence demand, revenue, and consumer choice, especially where service quality is difficult to observe before purchase (Chevalier and Mayzlin, 2006; Luca, 2016). For small and medium-sized enterprises, however, the managerial problem is not simply whether reviews matter. The harder issue is how to transform dispersed, multilingual, and unevenly written comments into a repeatable monitoring routine that can inform concrete service decisions.

Existing research offers strong evidence on the economic effects of review ratings and textual feedback, while the technical literature provides many sentiment-analysis tools. A gap remains between these streams. Academic studies often treat online-review analytics as an explanatory instrument, whereas practical scripts and platform documentation rarely connect data collection, sentiment scoring, sampling limits, and managerial interpretation within a reproducible SME-oriented workflow. This paper addresses that gap by positioning review listening as a lightweight business-analytics and decision-support process rather than as a one-off technical exercise.

The proposed workflow automates three tasks: retrieving recent Google Maps reviews for a known business location, retaining translated English text available through platform metadata, and producing reproducible sentiment and text-salience outputs. The empirical demonstration uses Klass Wagen Sibiu, an auto-rental location, and shows how 200 translated Google reviews can be converted into a concise monthly monitoring file. The analysis is not designed to adjudicate the accuracy of individual customer claims. Its purpose is to identify recurring friction points, protect strengths that customers repeatedly mention, and make review monitoring feasible for resource-constrained owners.

The contribution is therefore applied but still academic. It integrates literature on online reviews as market signals, text-rating discrepancy, sentiment analysis, and self-selection bias into a transparent pipeline that can be replicated, audited, and adapted. In doing so, the study clarifies the boundary between a useful managerial pulse and stronger inferential claims that would require validation, benchmarking, or longitudinal evidence.

2. LITERATURE REVIEW

2.1. Reviews as economic signals

Online reviews reduce information asymmetry by making prior customer experience visible to prospective buyers. In early e-commerce, consumer book reviews affected relative sales across online retailers (Chevalier and Mayzlin, 2006). In local services, Yelp ratings have been associated with revenue effects for independent restaurants (Luca, 2016), and discontinuities around rounded ratings can change demand near visible thresholds (Anderson and Magruder, 2012). More recent work also shows that review systems are not neutral mirrors of service quality because text, ratings, platform design, and reviewer self-selection interact. Text-rating review discrepancy is especially relevant for analytics because written comments and numerical stars do not always convey the same evaluation (Almansour *et al.*, 2022).

2.2. Text analytics for user-generated reviews

Sentiment-analysis approaches can be grouped into three broad families: lexicon-based and rule-based methods, traditional supervised machine-learning classifiers, and transformer-based models. Lexicon-based tools score text through dictionaries and handcrafted rules; they are transparent, inexpensive to run, and easy to explain to non-specialists. Supervised machine-learning models can capture richer patterns, but they normally require labelled data, feature engineering, and validation on a relevant domain. Transformer-based models, developed after the attention architecture and language-model pretraining, can deliver stronger contextual representations but add computational, interpretability, and maintenance costs (Vaswani *et al.*, 2017; Devlin *et al.*, 2019; Bordoloi and Biswas, 2023). For SMEs, this trade-off matters. A model that is marginally more accurate but opaque, costly, or difficult to rerun may be less useful for routine monitoring than a transparent baseline.

VADER fits the present purpose because it was designed for short, informal, socially expressed text and produces interpretable scores without labelled training data (Hutto and Gilbert, 2014). Its use here is not a claim that lexicon-based methods are universally superior. Rather, VADER is treated as a low-cost baseline suitable for recurring descriptive monitoring, with the explicit limitation that sarcasm, domain-specific expressions, and translated phrasing may require later validation or comparison with a transformer model.

Recent hospitality and restaurant applications further show that online review text can be converted into service dimensions and sentiment ind icators that support managerial interpretation, provided that the dataset, platform, language choices, and coding rules are made explicit (Nguyen and Dao, 2024).

2.3. Language and translation

Real-world businesses receive reviews in multiple languages. Automatic translation helps create a common analysis layer, but it can alter polarity, weaken idioms, and change affective nuance (Balahur and Turchi, 2014; Mohammad *et al.*, 2016). Using platform-provided translated text is still managerially relevant because this is often the text visible to users browsing in an English interface. The analytical obligation is therefore to document translation provenance and avoid treating translated sentiment as a direct equivalent of the original expression.

2.4. Platforms, access, and sampling

Access pathways also shape what can be analyzed. Google Places API exposes only a limited review sample per request, while aggregator services can support broader collection through paginated public review pages (SerpApi; Google Maps Platform, 2025). Regardless of access path, online reviews are affected by self-selection, polarity, and platform effects. Early or extreme reviewers may not represent the full customer base, and review distributions often overrepresent highly positive or strongly negative experiences (Li and Hitt, 2008; Schoenmueller *et al.*, 2020). Consequently, review analytics should be interpreted as directional evidence for managerial attention, not as an unbiased estimate of population-level satisfaction.

2.5. Visualization

Word clouds remain popular for rapid reconnaissance, but they can encourage superficial interpretation because they discard syntax, negation, and context (Harris, 2011). In this study, the word cloud is used only as an exploratory pointer to recurring vocabulary. It is interpreted together with sentiment labels, star ratings, and representative snippets rather than as independent evidence.

3. METHODOLOGY

The workflow follows the logic of CRISP-DM by linking business understanding, data preparation, modelling, and evaluation, but it is intentionally simplified for a small-business context (Wirth and Hipp, 2000). The pipeline has two scripts: one retrieves translated Google review text for a specific business location, and the other produces sentiment outputs and an exploratory vocabulary visualization. This design prioritizes transparency, repeatability, and low maintenance over methodological complexity.

3.1. Data collection with SerpApi

Data were collected through SerpApi Google Maps Reviews engine for the place identifier associated with Klass Wagen Sibiu. The request used an English interface setting and sorted reviews by recency. Reviews were retained only when the response contained a translated English text field. The collection was capped at 200 translated reviews, a pragmatic threshold chosen to keep the workflow lightweight while still providing enough observations for descriptive monitoring, frequency patterns, and review examples. The cap should not be read as a statistically representative sample of all customers.

The selection criteria were therefore explicit: one business location, recent Google review pages returned by the configured query, translated English text available in the extracted snippet, and a maximum of 200 retained observations. This procedure supports replication because the same place identifier, language setting, sorting rule, and cap can be rerun. It also creates identifiable constraints: untranslated reviews are excluded, the platform ordering and availability rules influence the sample, and the resulting corpus reflects voluntary reviewers rather than all customers.

3.2. Sentiment analysis with VADER

For sentiment scoring, the analysis applies VADER to the translated English text and stores per-review scores, labels, and summary statistics. The standard compound-score thresholds are used: scores of 0.05 or higher are labelled positive, scores of -0.05 or lower are labelled negative, and values between these thresholds are labelled neutral (Hutto and Gilbert, 2014). These thresholds provide a reproducible descriptive baseline. They are not treated as validated classification boundaries for this specific business domain.

3.3. Word cloud

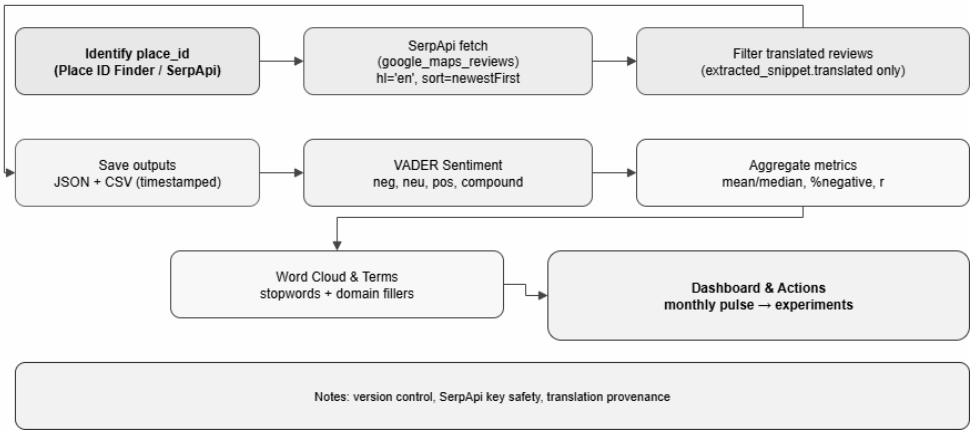
The word cloud is generated after removing generic stopwords, brand and location terms, and unavoidable industry words such as car, vehicle, and rental. The goal is to surface

recurring service and process terms rather than predictable context words. Because the visualization is exploratory, the interpretation remains tied to the sentiment table and manually inspected examples.

3.4. Reproducibility, versions, and environment

The scripts retain fixed inputs, package dependencies, thresholds, and output names to reduce configuration drift between runs. Timestamps are kept as returned by SerpApi, and no stochastic modelling is introduced. This makes the workflow auditable, although it also means that any change in platform access, translation availability, or stopword settings should be documented before comparing monthly outputs.

3.5. Workflow scheme



Source: own creation

Figure no. 1 – Workflow scheme

4. RESULTS

4.1. Dataset overview

The collection script retrieved 200 reviews containing translated English text. Reviews span multiple months and languages of origin, but the analysis is restricted to the translated text stored in the English text field. The business Google summary, provided by SerpApi response at the time of collection, indicated a rating of 4.7 across 1,683 total reviews.

4.2. Sentiment distribution and summary statistics

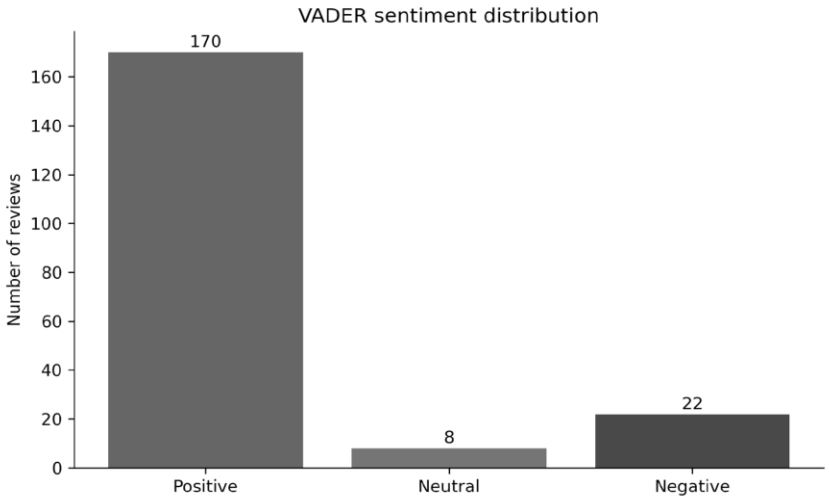
Table no. 1 summarizes the VADER outcomes from sentiment_summary.json.

Table no. 1 – Sentiment summary (translated texts, n = 200)

Metric	Value
Positive (compound ≥ 0.05)	170 (85%)
Neutral (–0.05 < compound < 0.05)	8 (4%)
Negative (compound ≤ –0.05)	22 (11%)
Mean compound	0.5377
Median compound	0.7328
Pearson correlation: star rating vs VADER compound	0.8302

Source: own creation

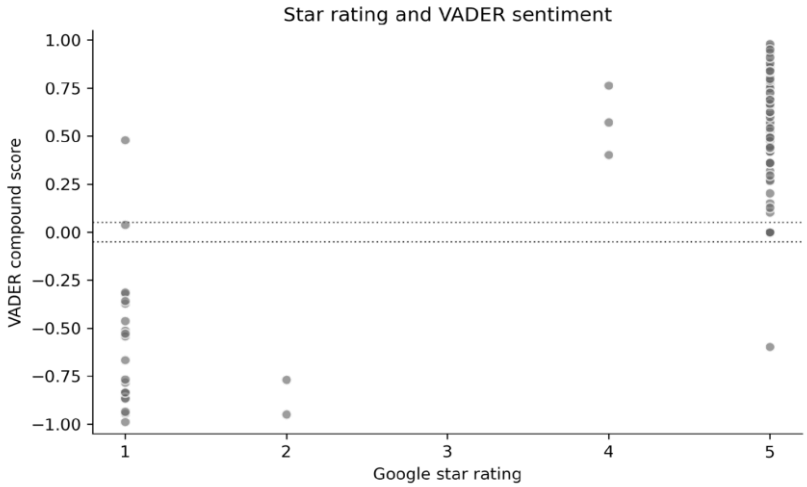
Two observations stand out. First, the distribution is strongly skewed positive. Second, VADER compound scores are highly associated with numeric ratings, r approximately 0.83. This association should be read cautiously. It suggests that the sentiment scores and star ratings move in a similar direction in this corpus, but it does not demonstrate classification accuracy. A stronger validation claim would require manually coded labels, a confusion matrix, agreement statistics, or a comparison with another sentiment model.



Source: own creation

Figure no. 2 – VADER sentiment label distribution

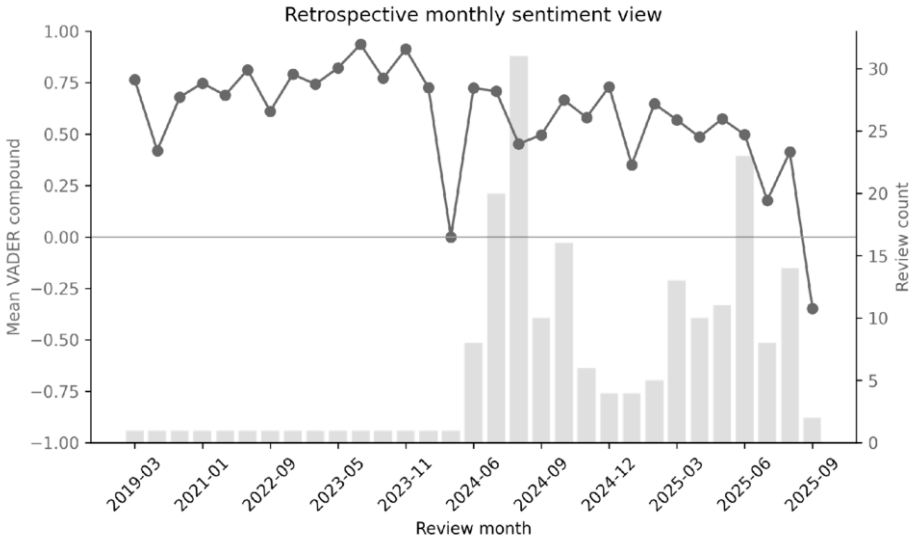
Figure no. 2 presents the distribution of VADER sentiment labels. The skew toward positive reviews is visually clear: 170 reviews are classified as positive, 8 as neutral, and 22 as negative. This distribution supports the descriptive claim that the corpus is broadly favorable, while still leaving a visible minority of negative cases for operational diagnosis.



Source: own creation

Figure no. 3 – Google star rating and VADER compound score

Figure no. 3 plots Google star ratings against VADER compound scores. Most 5-star reviews are located above the positive threshold, while most 1-star reviews fall below the negative threshold. A small number of discordant cases remain, including low-star reviews with non-negative text sentiment and high-star reviews with weaker textual polarity. These cases show why the pipeline should preserve both rating and text indicators.



Source: own creation

Figure no. 4 – Retrospective monthly sentiment view

Figure no. 4 provides a retrospective time view based on the review timestamps. The line reports monthly mean VADER compound scores and the bars report the number of reviews available in each month. This view is descriptive because reviews were not collected prospectively at fixed monthly intervals, but it illustrates how the same workflow can support repeated monitoring once run on a regular schedule.

4.3. What positive and negative reviews discuss

A rapid read of high- and low-score examples helps triangulate the numbers.

- **Positive exemplars** (high compound and 5★): "*I highly recommend, everything was perfect, thanks to the entire team.*" and "*Fast service for pick up and drop off. Everything was easy and hassle free. The staff was very friendly and very helpful.*" Sentiment drivers include *speed (fast), staff (friendly/helpful), cleanliness (clean/new),* and *recommendation intent* ("recommend," "100%").
- **Negative exemplars** (low compound and 1–2★): "*They pressured us to buy very expensive insurance; otherwise a huge deposit and threats of high costs for any small scratch.*" and "*Card 'not working' unless purchasing on-site insurance; return required carwash.*" Here the drivers are *insurance pressure, deposit size/handling, perceived unfairness, card acceptance,* and *carwash/cleaning requirements*.

These themes align with broader literature showing that checkout frictions and uncertainty about liability often dominate negative word-of-mouth in services (Luca, 2016).

4.4. Word cloud

Source: own creation

Figure no. 5 – Word cloud

Our word cloud removes generic vocabulary (the library's stopwords), brand/location words (e.g., *Klasswagen/KW, Sibiu, Romania*), and industry head terms (*car, vehicle, rental*) so that more diagnostic language surfaces. In this rendering, **positive** clusters center on *friendly, helpful, recommend, fast, professional, clean/new*, while **negative** clusters surface *insurance, deposit, credit card, scratch/damage, wash/cleaning*. Because word clouds ignore negation and syntax, we validate these impressions against the per-review sentiment table and spot-read several examples (Mueller; Harris, 2011).

Practical takeaway for owners: the cloud directs attention to recurring friction points that are actionable: (1) how insurance/deposits are explained, (2) clarity about cleaning/return expectations, and (3) card requirements. Conversely, positive themes (*friendly staff, fast pickup/drop-off, transparent process*) are differentiators to protect and standardize.

4.5. Interpretation of the correlation with ratings

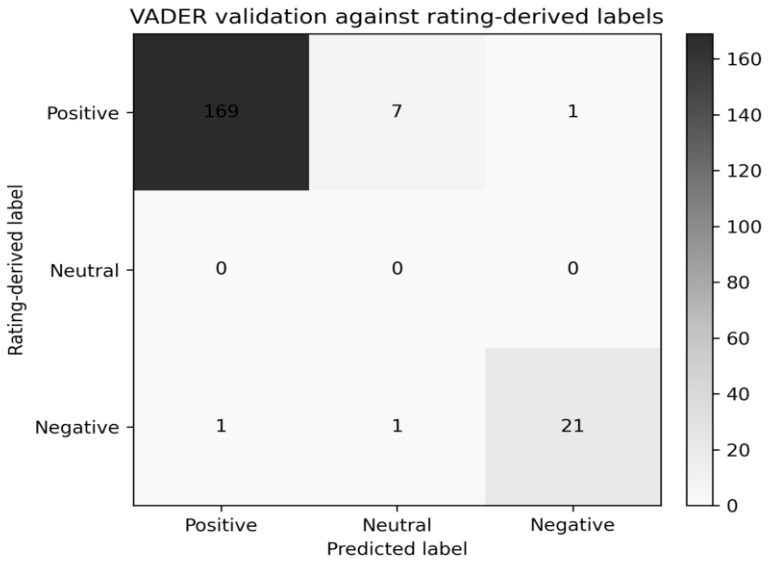
The strong correlation between VADER compound scores and star ratings is consistent with the many clearly positive and negative narratives in the corpus. At the same time, text-rating discrepancy research shows that written comments and numeric scores can diverge, so the correlation is best treated as descriptive convergence rather than proof that the model classifies sentiment correctly (Almansour *et al.*, 2022). Owners should therefore monitor both indicators: star distributions for high-level movement and review text for specific operational signals.

4.6. Rating-derived validation and benchmark comparison

To address the distinction between correlation and classification accuracy, an additional validation check was performed using rating-derived weak labels. Reviews with 4 or 5 stars were coded as positive, reviews with 1 or 2 stars were coded as negative, and 3-star reviews would be coded as neutral. In the present dataset, the rating-derived labels contain 177 positive and 23 negative cases, with no 3-star observations. This validation strategy is weaker than manual annotation, but it offers a transparent triangulation between numerical ratings and text-based sentiment labels.

Against these rating-derived labels, VADER correctly classified 190 of 200 reviews, yielding 95.0% accuracy and a weighted F1 score of 0.969. The confusion matrix shows that 169 positive cases and 21 negative cases were aligned with their rating-derived labels; the remaining disagreements mostly involve neutral VADER outputs for reviews whose star ratings were strongly positive or negative. Figure no. 4 reports this confusion matrix.

As a lightweight benchmark, a TF-IDF logistic-regression classifier was evaluated with stratified 5-fold cross-validation on the same translated review texts and rating-derived labels. This model reached 96.0% accuracy and a weighted F1 score of 0.960. The similarity between VADER and the supervised baseline supports the use of VADER as a transparent monitoring tool in this corpus, although the result should not be generalized beyond the present dataset without manual labels or additional cases.



Source: own creation

Figure no. 6 – VADER confusion matrix against rating-derived labels

4.7. Limitations and validity threats

Translation bias. The study analyzes only reviews with an English translation field. Automatic translation can alter sentiment by changing idioms, intensity, or polarity (Balahur and Turchi, 2014; Mohammad *et al.*, 2016). The translated text remains relevant for managerial monitoring because it approximates what many English-interface users see, but it is not equivalent to expert translation or original-language sentiment analysis.

Sampling and self-selection bias. Google review writers are volunteers, not a random sample of all customers. Customers with unusually positive or negative experiences may be more likely to post, and platform-level polarity can produce review distributions that differ from the underlying customer population (Li and Hitt, 2008; Schoenmueller *et al.*, 2020). The single-location design strengthens operational specificity but limits generalizability.

Platform and access bias. The corpus depends on the reviews made available through the platform interface and on the ordering, language, and pagination choices used during collection. These constraints may affect which reviews enter the dataset. The analysis therefore supports descriptive monitoring and issue detection rather than causal claims about service quality.

Method bias. VADER is a transparent lexicon-based tool, but it can miss sarcasm, domain-specific meanings, mixed sentiment, and complaints written in polite language (Hutto and Gilbert, 2014). Word clouds are similarly limited because they ignore grammar and negation (Harris, 2011). These outputs should trigger closer reading and, where decisions are high stakes, additional validation.

Validation constraint. The validation uses star ratings as weak labels rather than independent human annotation. This approach is useful for checking whether text sentiment

and rating-derived polarity broadly align, but it cannot replace manual coding because stars and text may express different aspects of customer experience.

Benchmark constraint. The TF-IDF logistic-regression comparison is a lightweight baseline trained and tested on rating-derived labels. It shows that a simple supervised model performs similarly to VADER on this corpus, but it does not establish superiority over transformer-based models or domain-specific human coding.

5. A REPLICABLE BLUEPRINT FOR ENTREPRENEURS

This section distills the demonstration into a recipe any owner can reuse.

1. **Identify your Google Maps place_id.** You can obtain it via Google's Place ID Finder or SerpApi's tooling. Keep it in a safe configuration file.
2. **Collect reviews monthly.** Run serpapi_klasswagen_reviews.py with your key and place id. Use hl="en" and keep only reviews with extracted_snippet.translated to maintain an English corpus. Save JSON + CSV with a date stamp (e.g., myplace_2025-09_en.json).
3. **Analyze sentiment.** Run analyze_sentiment.py to produce per-review VADER scores and the monthly summary. Append new rows to a long-format store (e.g., a single CSV partitioned by month) to track trends.
4. **Skim the word cloud.** Use it as a pointer to investigate a concrete policy area (e.g., *insurance explanation*), then read 10–20 representative reviews to decide on an action.
5. **Make a small change; measure again next month.** Examples: simplify deposit signage at pickup; add a pre-arrival email explaining credit card requirements; adopt a clearer cleaning policy. Look for a drop in negative share and a shift in word-cloud salience.
6. Document versions and assumptions. Keep package versions, thresholds, stopword changes, anonymized input files, processed outputs, and generated figures in the reproducibility package so a given month can be audited later.

6. TABLES AND EXHIBITS

Table no. 2 – Example positive and negative snippets with VADER outputs

Example (translated)	Stars	VADER compound	Label
"Fast service for pick up and drop off… staff was very friendly and very helpful. Highly recommend."	5	0.93	Positive
"Top-notch company. Very friendly staff, everything quick and easy."	5	0.84	Positive
"They pressured us to buy very expensive insurance… deposit threats for any small scratch."	1	–0.51	Negative
"Card 'not working' unless we bought on-site insurance; return required a carwash."	1	–0.95	Negative

Note: Examples paraphrased from the dataset to preserve privacy and focus on themes.

Source: own creation

Table 3. Operational dashboard metrics an owner can track monthly

Metric	Definition	Why it matters
% Negative	Negative labels ÷ total	Early warning; drill into themes
Mean compound	Average VADER score	High-level tone index
Rating–sentiment r	Pearson r between stars and compound	Checks method stability
Top negative bigrams	E.g., "additional insurance," "credit card," "car wash"	Turns themes into actions
Response lag	Average time to owner reply (if tracked)	Service recovery KPI

Source: own creation

7. DISCUSSION

The managerial value of the pipeline lies in a four-step monitoring routine: collect recent reviews, score the translated text, interpret recurring positive and negative themes, and act on a small number of service frictions in the next operating cycle. For the case analyzed here, the most salient negative themes concern insurance explanations, deposits, credit-card requirements, scratches or damage, and return cleaning. The most salient positive themes concern friendly staff, fast pickup and drop-off, clean vehicles, and recommendation intent. This structure turns descriptive analytics into an operational checklist without overstating the precision of the model.

From a research perspective, the study should be read as an applied decision-support blueprint. It does not introduce a new sentiment algorithm and does not claim population-level representativeness. Its contribution is to connect a replicable data-collection routine, a transparent sentiment baseline, and explicit validity boundaries in a format that can be reused by SMEs and audited by researchers. Future extensions can test whether the same workflow remains stable across locations, industries, languages, and monitoring periods.

8. CONCLUSION

The study shows that small firms can transform public review text into a repeatable monitoring routine without building complex analytics infrastructure. By combining SerpApi collection, VADER sentiment scoring, and carefully limited exploratory visualization, the workflow produces a monthly pulse of customer experience that is understandable to non-specialists and reproducible by a single analyst.

The findings support practical attention to recurring service frictions rather than broad causal claims. In the case analyzed, positive reviews emphasize speed, staff helpfulness, cleanliness, and recommendation intent, while negative reviews concentrate around insurance, deposits, credit-card requirements, damage disputes, and cleaning expectations. These patterns can guide targeted operational adjustments, especially clearer pre-arrival communication and more consistent explanation of return conditions.

The boundaries of the pipeline are equally important. It cannot prove that service quality changed, cannot remove self-selection from voluntary reviews, and cannot guarantee sentiment accuracy for translated or ambiguous text. Its value is iterative: run the same workflow regularly, compare the same indicators over time, read representative comments behind the metrics, make small service changes, and reassess whether the negative share and recurring friction vocabulary decline.

DATA AND CODE AVAILABILITY

An anonymized reproducibility package has been uploaded to GitHub and is available at the following repository address: https://github.com/MaraAlexandru/listening-at-scale-google-reviews-vader. The repository contains the translated review input file, sentiment-analysis scripts, processed outputs, validation files, and generated figures. It excludes API keys, direct user identifiers, Google profile links, contributor IDs, review links, thumbnails, and raw API metadata.

ORCID

Marian Pompiliu Cristescu https://orcid.org/0000-0003-3638-4379
Dumitru Alexandru Mara https://orcid.org/0000-0002-8898-2170
Ana-Maria Constantinescu https://orcid.org/0009-0004-2400-4409
Ioana Petrea https://orcid.org/0009-0009-3600-7857
Ana Englitera Visan https://orcid.org/0009-0002-1337-192X

References

Almansour, A., Alotaibi, R., & Alharbi, H. (2022). Text-rating review discrepancy (TRRD): An integrative review and implications for research. *8*. http://dx.doi.org/10.1186/s43093-022-00114-y

Anderson, M., & Magruder, J. (2012). Learning from the crowd: Regression discontinuity estimates of the effects of an online review database. *Economic Journal (London), 122*(563), 957–989. http://dx.doi.org/10.1111/j.1468-0297.2012.02512.x

Balahur, A., & Turchi, M. (2014). Comparative experiments using supervised learning and machine translation for multilingual sentiment analysis. *Computer Speech & Language, 28*(1), 56–75. http://dx.doi.org/10.1016/j.csl.2013.03.004

Bordoloi, M., & Biswas, S. K. (2023). Sentiment analysis: A survey on design framework, applications and future scopes. *Artificial Intelligence Review, 56*(1), 12505–12560. http://dx.doi.org/10.1007/s10462-023-10442-2

Chevalier, J. A., & Mayzlin, D. (2006). The effect of word of mouth on sales: Online book reviews. *JMR, Journal of Marketing Research, 43*(3), 345–354. http://dx.doi.org/10.1509/jmkr.43.3.345

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, Minneapolis, Minnesota.

Google Maps Platform. (2025). Place Details.

Harris, J. (2011). Word clouds considered harmful. *Nieman Lab.* Retrieved from https://www.niemanlab.org/2011/10/word-clouds-considered-harmful/

Hutto, C. J., & Gilbert, E. (2014). *VADER: A parsimonious rule-based model for sentiment analysis of social media text.* Paper presented at the Proceedings of the International AAAI Conference on Web and Social Media.

Li, X., & Hitt, L. M. (2008). Self-selection and information role of online product reviews. *Information Systems Research, 19*(4), 456–474. http://dx.doi.org/10.1287/isre.1070.0154

Luca, M. (2016). *Reviews, reputation, and revenue: The case of Yelp.com* (Harvard Business School Working Paper No. 12-016). Retrieved from Boston, MA: https://www.hbs.edu/ris/Publication%20Files/12-016_a7e4a5a2-03f9-490d-b093-8f951238dba2.pdf

Mohammad, S. M., Salameh, M., & Kiritchenko, S. (2016). How translation alters sentiment. *Journal of Artificial Intelligence Research, 55*, 95–130. http://dx.doi.org/10.1613/jair.4787

Mueller, A. wordcloud: A little word cloud generator in Python: wordcloud. Retrieved from https://amueller.github.io/word_cloud/

Nguyen, H. L., & Dao, T. H. (2024). Text mining: Sentiment analysis of reviews on TripAdvisor for Vietnam's Michelin-starred and selected restaurants in the 2023 Michelin Guide. *11*(2), 131–148. http://dx.doi.org/10.33851/JMIS.2024.11.2.131

Schoenmueller, V., Netzer, O., & Stahl, F. (2020). The polarity of online reviews: Prevalence, drivers and implications. *JMR, Journal of Marketing Research, 57*(5), 853–877. http://dx.doi.org/10.1177/0022243720941832

SerpApi. Google Maps Reviews API. *SerpApi.* Retrieved from https://serpapi.com/google-maps-reviews-api

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). *Attention is All you Need*. Paper presented at the Advances in Neural Information Processing Systems. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Wirth, R., & Hipp, J. (2000). *CRISP-DM: Towards a standard process model for data mining*. Paper presented at the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining. https://www.cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf

ANNEX A.1 ANALYZE_SENTIMENT.PY

```
#!/usr/bin/env python3
"""
Analyze sentiment on translated Google reviews (VADER) and build a word cloud.

Input (hardcoded):
  klasswagen_translated_only.json
    {
     "place_info": {...},
     "reviews": [
       {
        "extracted_snippet": {
          "original": "...",
          "translated": "ENGLISH TEXT HERE"
        },
        "rating": 5.0,
        "iso_date": "2025-09-01T07:10:48Z",
        "review_id": "..."
       }, ...
     ]
    }

Outputs (hardcoded):
  sentiment_by_review.csv     # one row per review with VADER scores + label
  sentiment_summary.json      # averages + counts
  wordcloud_translated.png    # word cloud from translated text (stopwords removed)
```

```
Install once:
    pip install vaderSentiment wordcloud matplotlib pandas

Note:
    We score sentiment on the English *translated* text only and do NOT filter it with stopwords.
    Stopwords are applied only to the wordcloud (to “ignore filler words” visually).
"""

from __future__ import annotations

import json
import re
from pathlib import Path
from typing import Dict, List

import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt


# ---------------------------- HARD-CODED PATHS ----------------------------
INPUT_JSON = Path("klasswagen_translated_only.json")
OUT_CSV = Path("sentiment_by_review.csv")
OUT_SUMMARY_JSON = Path("sentiment_summary.json")
OUT_WORDCLOUD = Path("wordcloud_translated.png")


# ---------------------------- HELPERS -------------------------------------
def load_translated_reviews(path: Path) -> List[Dict]:
    data = json.loads(path.read_text(encoding="utf-8"))
    reviews = data.get("reviews", []) or []

    out = []
    for r in reviews:
        ex = r.get("extracted_snippet") or {}
        text_en = ex.get("translated")  # we saved only translated ones earlier
        if isinstance(text_en, str) and text_en.strip():
            out.append(
                {
                    "review_id": r.get("review_id"),
                    "iso_date": r.get("iso_date"),
                    "rating": r.get("rating"),
                    "text_en": text_en.strip(),
                }
            )
    return out


def vader_label(compound: float) -> str:
    # Standard VADER thresholds
```

```
    if compound >= 0.05:
        return "positive"
    elif compound <= -0.05:
        return "negative"
    return "neutral"


def build_wordcloud(texts: List[str], out_path: Path) -> None:
    # Start from WordCloud's built-in STOPWORDS and extend with domain fillers
    extra_stop = {
        # brand / domain-specific fillers that add little meaning to the cloud
        "klasswagen", "klass", "wagen", "kw",
        # recurring codes / tokens
        "sbz", "sbz-", "sbz_", "sbz - ",
        # generic business/location fillers
        "romania", "sibiu", "bucharest", "cluj", "napoca", "airport",
        "company", "team", "services", "service",
        # industry-common words that would dominate the cloud
        "car", "cars", "rental", "rent", "vehicle", "vehicles",
        # politeness fillers
        "hello", "hi", "dear", "thanks", "thank", "please",
    }
    stopwords = STOPWORDS.union(extra_stop)

    # Clean: drop URLs, collapse whitespace. (We keep punctuation; WordCloud tokenizes.)
    cleaned_texts = []
    url_re = re.compile(r"https - ://\S+")
    code_re = re.compile(r"\b( - :sbz)[\w\-]*\b", flags=re.IGNORECASE)
    for t in texts:
        t2 = url_re.sub(" ", t)
        t2 = code_re.sub(" ", t2)
        t2 = re.sub(r"\s+", " ", t2)
        cleaned_texts.append(t2.strip())

    blob = "\n".join(cleaned_texts)

    wc = WordCloud(
        width=1800,
        height=1000,
        background_color="white",
        stopwords=stopwords,
        collocations=True,   # keep meaningful bigrams if frequent
        max_words=400,
    ).generate(blob)

    # Save to disk
    plt.figure(figsize=(18, 10))
    plt.imshow(wc, interpolation="bilinear")
    plt.axis("off")
    plt.tight_layout(pad=0)
    plt.savefig(out_path, dpi=180)
```

```
    plt.close()

# ---------------------------- MAIN ----------------------------------------
def main() -> None:
    if not INPUT_JSON.exists():
        raise SystemExit(f"Input not found: {INPUT_JSON.resolve()}")

    rows = load_translated_reviews(INPUT_JSON)
    if not rows:
        raise SystemExit("No translated reviews found in the input JSON.")
    # VADER sentiment
    analyzer = SentimentIntensityAnalyzer()
    scored = []
    for r in rows:
        s = analyzer.polarity_scores(r["text_en"])
        compound = s["compound"]
        scored.append(
            {
                "review_id": r["review_id"],
                "iso_date": r["iso_date"],
                "rating": r["rating"],
                "text_en": r["text_en"],
                "vader_neg": s["neg"],
                "vader_neu": s["neu"],
                "vader_pos": s["pos"],
                "vader_compound": compound,
                "sentiment_label": vader_label(compound),
            }
        )

    df = pd.DataFrame(scored)
    df.to_csv(OUT_CSV, index=False, encoding="utf-8")

    # Summary
    summary = {
        "n_reviews": int(df.shape[0]),
        "vader_compound_mean": float(df["vader_compound"].mean()),
        "vader_compound_median": float(df["vader_compound"].median()),
        "counts": {
            "positive": int((df["sentiment_label"] == "positive").sum()),
            "neutral": int((df["sentiment_label"] == "neutral").sum()),
            "negative": int((df["sentiment_label"] == "negative").sum()),
        },
        # (Optional) Rating correlation with sentiment if ratings exist
        "rating_vs_compound_correlation": (
            float(df["rating"].corr(df["vader_compound"])) if "rating" in df and
df["rating"].notna().any() else None
        ),
    }
    OUT_SUMMARY_JSON.write_text(json.dumps(summary, indent=2), encoding="utf-8")
```

```
    # Wordcloud from translated text (ignoring stopwords/fillers)
    build_wordcloud(df["text_en"].tolist(), OUT_WORDCLOUD)

    print(f"Done.\n- Wrote {OUT_CSV}\n- Wrote {OUT_SUMMARY_JSON}\n- Wrote {OUT_WORDCLOUD}")


if __name__ == "__main__":
    main()
```

ANNEX A.2 SERPAPI_KLASSWAGEN_REVIEWS.PY

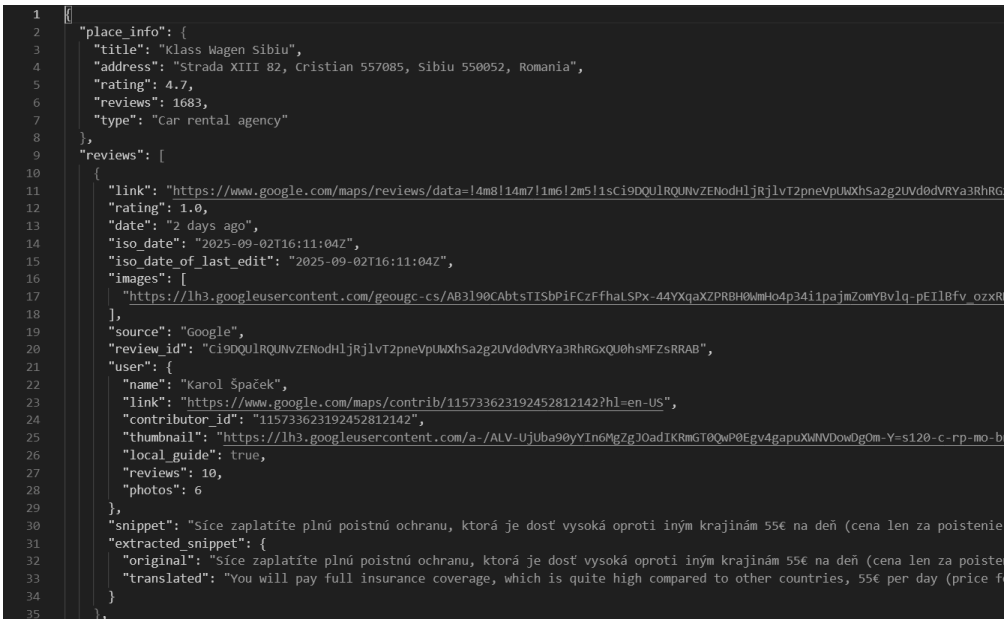
```
#!/usr/bin/env python3
"""
Fetch up to 200 *translated* Google Maps reviews for Klass Wagen via SerpApi
 -  ONLY rows that have `extracted_snippet.translated` are saved.

Hardcoded:
- SerpApi API key
- Google Maps place_id
- Output file names
- Sort order and language context

Requirements:
    pip install google-search-results

NOTE: Your API key is embedded in plaintext. Share/commit with care.
"""

from __future__ import annotations

import csv
import json
import time
import os
from typing import Any, Dict, List, Optional

from serpapi import GoogleSearch

# -------------------- HARD-CODED SETTINGS --------------------
API_KEY   = os.getenv("SERPAPI_API_KEY", "REDACTED")  # ← MASKED
PLACE_ID  = "ChIJQbiBR6JCTEcRllC2V-CB2uI"  # Klass Wagen Sibiu
HL        = "en"              # request English UI/context
SORT_BY   = "newestFirst"       # qualityScore | newestFirst | ratingHigh | ratingLow
MAX_TRANSLATED = 200          # stop once we have this many translated reviews
SLEEP_BETWEEN_PAGES = 0.35      # be polite between page requests
OUT_PREFIX = "klasswagen_translated_only"  # outputs: .json and .csv

# -------------------------------------------------------------
```

```
def fetch_translated_only() -> Dict[str, Any]:
    """
    Use SerpApi Google Maps Reviews API to fetch reviews for PLACE_ID,
    paginate with next_page_token, and keep ONLY those with a translated text
    at review['extracted_snippet']['translated'].
    Returns dict with 'place_info' and 'reviews' (translated-only).
    """
    params = {
        "engine": "google_maps_reviews",
        "api_key": API_KEY,
        "place_id": PLACE_ID,
        "sort_by": SORT_BY,
        "hl": HL,
    }
    search = GoogleSearch(params)

    place_info: Optional[Dict[str, Any]] = None
    translated_reviews: List[Dict[str, Any]] = []
    seen_ids: set[str] = set()

    while len(translated_reviews) < MAX_TRANSLATED:
        result = search.get_dict()

        # Basic error handling
        if "error" in result:
            raise RuntimeError(f"SerpApi error: {result['error']}")

        if place_info is None:
            place_info = result.get("place_info")

        page_reviews = result.get("reviews", []) or []
        for r in page_reviews:
            rid = r.get("review_id")
            if rid and rid in seen_ids:
                continue

            ex = r.get("extracted_snippet") or {}
            translated_text = ex.get("translated")

            # Keep ONLY reviews that have a translated version
            if isinstance(translated_text, str) and translated_text.strip():
                translated_reviews.append(r)
                if rid:
                    seen_ids.add(rid)
                if len(translated_reviews) >= MAX_TRANSLATED:
                    break

        if len(translated_reviews) >= MAX_TRANSLATED:
            break

        # Pagination: use next_page_token for the next batch; request up to 20 after first page
```

```
        serp_pagination = result.get("serpapi_pagination") or {}
        next_token = result.get("next_page_token") or serp_pagination.get("next_page_token")
        if not next_token:
            break  # no more pages available

        search.params_dict.update({
            "next_page_token": next_token,
            "num": 20,  # allowed after the first page
        })
        time.sleep(SLEEP_BETWEEN_PAGES)

    return {
        "place_info": place_info,
        "reviews": translated_reviews[:MAX_TRANSLATED],
    }

def flatten_review_translated(r: Dict[str, Any]) -> Dict[str, Any]:
    """
    Flatten fields for CSV, putting the translated English text into `text_en`.
    Also include `original_text` for reference.
    """
    user = r.get("user") or {}
    ex = r.get("extracted_snippet") or {}
    response = r.get("response") or {}
    rex = response.get("extracted_snippet") or {}

    return {
        "review_id": r.get("review_id"),
        "rating": r.get("rating"),
        "date": r.get("date"),
        "iso_date": r.get("iso_date"),
        "likes": r.get("likes"),
        "text_en": ex.get("translated"),
        "original_text": ex.get("original"),
        "review_link": r.get("link"),
        "user_name": user.get("name"),
        "user_link": user.get("link"),
        "user_contributor_id": user.get("contributor_id"),
        "user_local_guide": user.get("local_guide"),
        "user_reviews_count": user.get("reviews"),
        "user_photos_count": user.get("photos"),
        # Optional: include company reply (translated if present)
        "reply_text_en": rex.get("translated"),
        "reply_original_text": rex.get("original"),
        "reply_iso_date": (response.get("iso_date") or response.get("iso_date_of_last_edit")),
    }

def write_outputs(payload: Dict[str, Any]) -> None:
    """Write JSON + CSV with hardcoded filenames."""
    # JSON (translated-only)
    with open(f"{OUT_PREFIX}.json", "w", encoding="utf-8") as f:
```

```
        json.dump(payload, f, ensure_ascii=False, indent=2)

    # CSV (translated-only)
    rows = [flatten_review_translated(r) for r in (payload.get("reviews") or [])]
    fieldnames = list(rows[0].keys()) if rows else list(flatten_review_translated({}).keys())
    with open(f"{OUT_PREFIX}.csv", "w", newline="", encoding="utf-8") as f:
        writer = csv.DictWriter(f, fieldnames=fieldnames)
        writer.writeheader()
        writer.writerows(rows)

def main() -> None:
    payload = fetch_translated_only()
    write_outputs(payload)

    place = (payload.get("place_info") or {}).get("title")
    print(f"Place: {place or 'Unknown'}")
    print(f"Translated reviews saved: {len(payload.get('reviews', []))}")
    print(f"Wrote: {OUT_PREFIX}.json and {OUT_PREFIX}.csv")

if __name__ == "__main__":
    main()
```

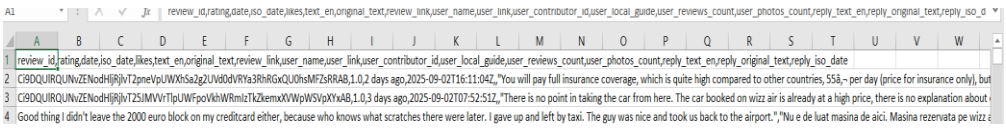
ANNEX A.3 ARTIFACT PLACEHOLDERS (PASTE SCREENSHOTS)




```
{
  "place_info": {
    "title": "Klass Wagen Sibiu",
    "address": "Strada XIII 82, Cristian 557085, Sibiu 550052, Romania",
    "rating": 4.7,
    "reviews": 1683,
    "type": "Car rental agency"
  },
  "reviews": [
    {
      "link": "https://www.google.com/maps/reviews/data=!4m8!14m7!1m6!2m5!1sCi9DQUlRQUNvZENodHljRjlvT2pneVpUWXhSa2g2UVd0dVRYa3RhRG
      "rating": 1.0,
      "date": "2 days ago",
      "iso_date": "2025-09-02T16:11:04Z",
      "iso_date_of_last_edit": "2025-09-02T16:11:04Z",
      "images": [
        "https://lh3.googleusercontent.com/geougc-cs/AB3l90CAbtsTISbPiFCzFfhaLSPx-44YXqaXZPRBH0WmHo4p34i1pajmZomYBvlq-pEIlBfv_ozxR
      ],
      "source": "Google",
      "review_id": "Ci9DQUlRQUNvZENodHljRjlvT2pneVpUWXhSa2g2UVd0dVRYa3RhRGxQU0hsMFZsRRAB",
      "user": {
        "name": "Karol Špaček",
        "link": "https://www.google.com/maps/contrib/115733623192452812142?hl=en-US",
        "contributor_id": "115733623192452812142",
        "thumbnail": "https://lh3.googleusercontent.com/a-/ALV-UjUba90yYIn6MgZgJOadIKRmGT0QwP0Egv4gapuXWNVDowDgOm-Y=s120-c-rp-mo-b
        "local_guide": true,
        "reviews": 10,
        "photos": 6
      },
      "snippet": "Síce zaplatíte plnú poistnú ochranu, ktorá je dosť vysoká oproti iným krajinám 55€ na deň (cena len za poistenie
      "extracted_snippet": {
        "original": "Síce zaplatíte plnú poistnú ochranu, ktorá je dosť vysoká oproti iným krajinám 55€ na deň (cena len za poiste
        "translated": "You will pay full insurance coverage, which is quite high compared to other countries, 55€ per day (price f
      }
    },
```

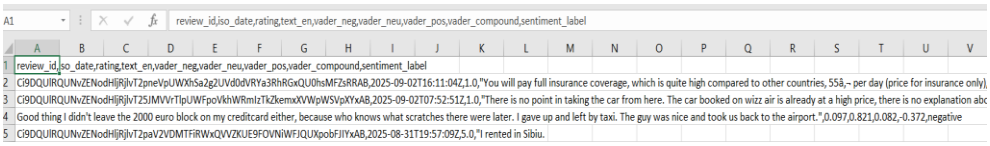

Source: own creation

Figure no. A1 – klasswagen_translated_only.json - place metadata and 200 translated reviews.



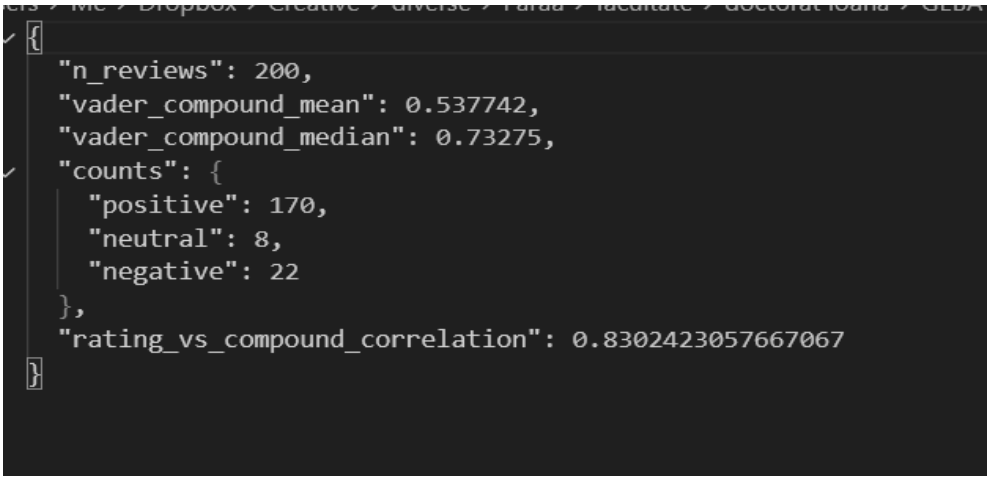
Source: own creation

Figure no. A2 – klasswagen_translated_only.csv - flattened fields including text_en



Source: own creation

Figure no. A3 – sentiment_by_review.csv - per-review VADER scores and labels



Source: own creation

Figure no. A4 – sentiment_summary.json - summary metrics (counts, mean/median, correlation)