SpotLite: A System for Personalized, Aspect-Based Restaurant Recommendation Near Tourist Destinations

Ting-Long Wei
tweI0542@usc.edu

Yu-Chen Lu
ylu74747@usc.edu

Sheng-Kuo Lin
slin7099@usc.edu

Shih-Hui Huang
shihhuih@usc.edu

Jingyi Xia
xiajingy@usc.edu

Suihan Gao
suihanga@usc.edu

University of Southern California, Los Angeles, CA, USA

Abstract

This report presents the ongoing development of the SpotLite project, a context-aware restaurant recommendation system leveraging natural language processing on user reviews. We describe the project goal, dataset collection and preprocessing, initial insights, and our concrete plan for final model implementation and evaluation.

1 Project Goal

Commercial platforms (e.g., Google Maps) exhibit three primary limitations for users with specific intents: (1) **Information Overload**, requiring manual review parsing to identify key attributes; (2) **Fragmented Keywords**, as semantically equivalent terms are not consolidated; and (3) **Limited Personalization**, with filters that lack support for multi-faceted queries.

SpotLite addresses these issues by ingesting a location URL and a structured preference set to output a ranked list of venues with summarized, aspect-level intelligence. The project's scope is confined to English-language reviews from the past year. The system will retrieve relevant English reviews from Google Maps, summarize them, and ultimately return the most suitable restaurants with a summary of their key features.

This report follows a complete project lifecycle, from the initial data acquisition strategy to the architectural design and model selection for the core NLP engine, concluding with end-to-end implementation details.

2 Dataset Details

2.1 Dataset Collection and Preprocessing

This section describes the dataset collection and preprocessing methodology for the SpotLite project. It outlines the data sources, structural schema, and filtering criteria for constructing an initial training corpus, which is geographically confined to restaurants within the Los Angeles metropolitan area. The data collection phase aims to establish a robust corpus for model training and evaluation.

2.1.1 Data Acquisition and Structure

We collect data by web-scraping Google Maps reviews of restaurants within our target area. The review data is supplemented with basic restaurant metadata obtained through the Google Places API, including the restaurant's name, address, operating hours, and overall rating. This supplementary information enables filtering by user distance and current operational status, ensuring that only open restaurants are recommended.

The scraping process produces a structured JSON object for each restaurant, containing an array of individual reviews. Each review entry includes several key fields essential for downstream analysis:

- **review_id:** Unique identifier for the review.
- **reviewer:** Name of the reviewer.
- **subtitle:** Supplementary details of the reviewer, such as "Local Guide" status or total review count.
- **stars:** Quantitative rating provided (e.g., 5 stars).
- **date:** Relative timestamp of the review (e.g., 2 weeks ago, a year ago).
- **text:** Full textual content of the review, serving as the primary input for our NLP models.
- **photo_urls:** Array of URLs for user-submitted photos, potentially used for multimodal extensions.

As shown in our sample data structure below, each review entry includes several key fields essential for downstream analysis.

```
{
  "review_id": "ChdDSUhNMG9nS0VJQ0
      FnTUNBaTZTeHB3RRAB",
  "reviewer": "chewie",
  "subtitle": "Local Guide    26
      reviews    6 photos",
  "stars": "5 stars",
  "date": "8 months ago",
  "text": "The best YGF location out
      there!!! Ingredient choices are
      awesome and CLEAN and much more
      than other places.",
  "photo_urls": [
    "https://lh3.googleusercontent.com
      /.../photo1=w1200-h900-p"
  ]
}
```

Figure 1: Example JSON structure for a restaurant review entry.

2.1.2 Filtering and Curation

The raw scraped data undergoes cleaning and filtering to ensure its suitability for model training. Our preprocessing pipeline involves several key stages:

1. **Initial Cleaning:** Manual inspection and automated cleaning remove irrelevant, malformed, or incomplete data entries.
2. **Restaurant-Level Filtering:** Retain only restaurants with at least 50 total reviews to ensure statistical reliability and mitigate data sparsity.
3. **Review-Level Filtering:** Retain only reviews posted within the past year, ensuring the dataset reflects up-to-date customer experiences.

After filtering, we compute **term frequency–inverse document frequency (TF–IDF)** scores over the textual reviews of each restaurant to extract its representative keywords.

For every restaurant, we treat all of its reviews as a single document and apply TF–IDF vectorization to identify the most distinctive terms that capture local aspects such as cuisine type, service quality, or ambiance descriptors. These extracted keywords serve two purposes: (1) they guide aspect-based sentiment analysis by highlighting salient thematic dimensions, and (2) they contribute to explainable, content-driven restaurant recommendations.

2.1.3 Target Corpus

Following this multi-stage preprocessing pipeline, we aim to construct a high-quality corpus comprising approximately 1,000 distinct restaurants from the Los Angeles area. This dataset will form the foundation for training and evaluating our aspect extraction, sentiment analysis, and summarization models.

3 Initial Insights

We implemented an end-to-end analysis pipeline that converts raw Google-style review JSON into actionable, human-readable insights for each venue. The pipeline emphasizes **traceability** (every claim is grounded in sentences), **low supervision** (no hand-curated lexicons), and **efficiency** (small, fast models). Below are steps that our project does to handle raw review data.

3.1 Preprocessing

Given a store's raw JSON, we normalize fields, split reviews into sentences, and de-duplicate near-identical sentences. The result is a compact corpus per store suitable for sentence-level inference:

```
{
  "store_id": "ygf_reviews",
  "sentences": [
    {"review_id": "...", "sentence": "
        The broth was silky and rich
        ."},
    {"review_id": "...", "sentence": "
        We waited about 20 minutes."},
    ...
  ]
}
```

Figure 2: Example of preprocessed store-level corpus.

3.2 Sentence-level sentiment

We score each sentence with a lightweight, pretrained sentiment model (DistilBERT SST-2), mapping probability to a signed score $s_i \in [-1, 1]$. For high-level tracking, we aggregate by store with a calibrated mean and label:

$$\text{score}_{\text{text}} = \frac{1}{N}\sum_i s_i, \qquad \text{label} \in \{\text{Positive}, \text{Neutral}, \text{Negative}\}. \tag{1}$$

(Optionally, star ratings can be blended as a weak prior when available.)

3.3 Aspect surface (unsupervised)

To move beyond fragile global n-grams, we build aspects by (i) encoding sentences with **MiniLM** embeddings, (ii) clustering them using **k-means**, and (iii) labeling each cluster with **class-TF-IDF** over n-grams (1–3) computed **within** the cluster (not corpus-wide). Each cluster c receives a polarity from the mean sentiment of its members:

$$\mu_c = \frac{1}{|c|} \sum_{i \in c} s_i, \tag{2}$$

and we rank clusters for display by

$$R_c = |\mu_c| \cdot \log(1 + |c|), \tag{3}$$

favoring well-supported, strongly opinionated themes. This approach naturally surfaces multi-word collocations such as *"silky broth"* or *"slow service"* while suppressing generic tokens.

3.4 Evidence selection and diversity

For each selected **Pro/Con** cluster, we keep 1–2 distinctive phrases and retrieve 1–2 **evidence sentences** from that cluster (positive examples for Pro, negative for Con). A greedy Jaccard filter ensures diversity, preventing multiple near-duplicates from crowding the evidence list.

3.5 Summary generation (grounded)

The final output per store contains (a) a short **paragraph** that weaves top pro/cons into fluent prose, (b) **pros/cons bullet points** (phrases), and (c) a small set of **verbatim quotes**.

3.6 Artifacts produced

The workflow emits interpretable JSONs at each stage:

- `*_sentences_scored.json` — per-sentence sentiment.
- `*_aspects.json` — clusters with sizes, mean scores, top phrases, and example sentences.
- `*_summary.json` — final paragraph, pros/cons lists, and evidence quotes; optional badge from the numeric roll-up.

3.7 Motivating results

On a real store sample, the system reliably extracts coherent **Pro** themes (e.g., *"silky broth", "tender chashu", "fresh ingredients"*) and **Con** themes (e.g., *"long waits (˜20 min)", "parking difficulty"*), with quotes that read naturally and are easy to audit. Compared with a naïve corpus n-gram approach, the cluster-based surface (embeddings → clusters → class-TF-IDF) substantially reduces noisy unigrams and increases multi-word specificity, while requiring no domain-specific lexicon. In the end, it can generate the short paragraph(s) of summary that human can read and quickly knows about this store sample.

4 Planned Work

We outline the remaining development and evaluation before the final report presentation (due November 25). Each week includes concrete deliverables to ensure consistent progress.

4.1 Week 1: Data Expansion

- Improve review crawler to fully support automatic pagination
- Scale dataset to ∼1,000 restaurants with complete metadata

Deliverables: Updated dataset (JSON), statistics report on review coverage

4.2 Week 2: NLP Refinement

- Enhance keyword post-processing (e.g., mapping "20 minutes" → "long wait time")
- Improve aspect clustering quality using contextual embeddings

Deliverables: Improved sentiment + keyword extraction module, qualitative error analysis

4.3 Week 3: Summarization & Ranking Integration

- Fine-tune a T5-based model for aspect-aware abstractive summaries
- Implement user-preference ranking pipeline (distance, rating, aspect scores)

Deliverables: End-to-end ranking system, example summaries and ranked outputs

4.4 Week 4: Deployment & Evaluation

- Develop a web-based interface for interactive recommendation
- Conduct user testing and measure relevance (NDCG@10) and task time

Deliverables: Web application prototype, user study report, evaluation metrics