Comprehensive Preprocessing Workflow Report

Tourism Sentiment Analysis Project

Team ScourgifyData

January 7, 2026

Abstract

This report provides an in-depth documentation of the data preprocessing pipeline established for the development of a robust sentiment analysis model tailored to the tourism sector. Starting from the massive Yelp Academic Dataset, we conducted a rigorous data selection process, evaluating five distinct sub-datasets before narrowing our focus to the Review and Business datasets. The workflow encompasses data integration, extensive cleaning, feature engineering, and a sophisticated synonym-based upsampling strategy to resolve severe class imbalance. The final output is a high-quality, balanced, and vectorized dataset optimized for machine learning applications.

Contents

1 Introduction

In the domain of Natural Language Processing (NLP), the quality of the input data is often more critical than the complexity of the model itself. This project aims to build a sentiment classifier capable of automatically categorizing tourism-related reviews into positive, neutral, and negative sentiments. To achieve this, we leveraged the Yelp Academic Dataset, a rich repository of user interactions and business information.

However, raw real-world data is rarely ready for immediate modeling. It contains noise, irrelevant information, missing values, and structural imbalances that can severely degrade model performance. This report details the end-to-end preprocessing workflow, highlighting the strategic decisions made to transform raw JSON data into a clean, numerical format. A key focus of this report is the rationale behind our data selection process—specifically, why we discarded the majority of the available datasets to focus exclusively on reviews—and how we addressed the challenge of class imbalance.

2 Data Acquisition and Selection Strategy

The foundation of this project is the Yelp Academic Dataset, a standard benchmark dataset for NLP and recommendation systems. Upon initial acquisition, the dataset consisted of five distinct JSON files, each capturing a different dimension of the user-business interaction ecosystem.

2.1 The Five Component Datasets

We began our analysis by exploring the contents and potential utility of each of the five available datasets:

1. **Business Dataset (`business.json`)**: Contains detailed information about local businesses, including location, categories, attributes (e.g., "Good for Kids"), and aggregate star ratings.

2. **Review Dataset (`review.json`)**: The core dataset containing full text reviews written by users, along with a star rating (1-5) and timestamps.

3. **User Dataset (`user.json`)**: Provides metadata about the users, such as their friend network, total review count, and average rating given.

4. **Check-in Dataset (`checkin.json`)**: Logs the timestamps of when users checked into businesses via the Yelp mobile app.

5. **Tip Dataset (`tip.json`)**: Contains short, quick tips left by users (e.g., "Try the secret menu"), which are distinct from full-length reviews.

2.2 Evaluation and Selection Rationale

Our primary objective was to classify sentiment based on textual content. This goal guided our selection process. We evaluated each dataset against criteria of relevance, information density, and redundancy.

2.2.1 Discarded Datasets

After a thorough preliminary analysis, we decided to discard four of the five datasets from the final modeling phase:

- **Business Dataset**: Although crucial for the initial filtering phase (to identify tourism-related entities), the business attributes themselves (e.g., location, hours) were not used as features in the sentiment analysis model. Once the relevant reviews were extracted via merging, this dataset was discarded to focus purely on textual content.

- **User Dataset**: While valuable for personalization or recommendation systems, user metadata (e.g., number of friends) does not inherently correlate with the *sentiment* of a specific text review. Including this data would add noise and dimensionality without contributing to the text classification task.

- **Check-in Dataset**: Check-in data is purely temporal and geographical. It indicates popularity or foot traffic but contains no textual information regarding customer satisfaction or sentiment. Therefore, it was deemed irrelevant for our model.
- **Tip Dataset**: Although tips contain text, they are typically very short, context-dependent, and lack the depth of full reviews. Furthermore, they often lack an associated star rating, making supervised learning difficult. We determined that the **Review Dataset** provided a far richer source of linguistic data.

2.2.2 Retained Datasets

We concluded that only one dataset was essential for the final modeling workflow:

- **Review Dataset**: This was selected as our **primary and sole data source** for training. It contains the actual text ("The room was dirty") and the ground-truth label (Star Rating), which are the direct inputs and targets for our model.

3 Data Integration and Filtering

Having selected the Review and Business datasets, the next step was to integrate them to create a unified, domain-specific dataset.

3.1 Filtering for Tourism

The raw Business dataset contains millions of businesses across diverse categories (e.g., Dentists, Mechanics, Restaurants). To isolate our target domain:

1. We scanned the `categories` column of the Business dataset.
2. We applied a filter to retain only businesses associated with tourism-related keywords.
3. This reduced the Business dataset to a subset of relevant entities.

3.2 Dataset Merging

We performed a **Left Join** operation merging the **Review Dataset** (left) with the filtered **Business Dataset** (right) on the unique key `business_id`.

- **Logic**: `Reviews.merge(Business, on='business_id', how='left')`
- **Outcome**: This operation attached business metadata to every review. Crucially, it allowed us to filter the reviews. Any review associated with a non-tourism business (where the merge resulted in null business data) was discarded.
- **Result**: A unified dataframe containing only reviews for tourism businesses.

4 Data Cleaning and Preprocessing

With the raw tourism dataset established, we initiated a multi-stage cleaning process to ensure data quality.

4.1 Feature Selection and Dimensionality Reduction

The merged dataset contained numerous columns that were unnecessary for text classification. To improve processing speed and reduce memory usage, we dropped the following:

- `business_id` and `review_id`: Unique identifiers with no predictive value.
- `categories`: No longer needed after the filtering step.
- `date`, `useful`, `funny`, `cool`: Metadata features not used in this specific text-based model.

We retained only two columns: `text` (the input) and `stars` (the target).

4.2 Handling Missing and Anomalous Data

- **Null Values**: We performed a check for NaN values. Any rows with missing text or ratings were dropped to maintain dataset integrity.
- **Empty Strings**: We converted the text column to string type and stripped leading/trailing whitespace.
- **Short Reviews**: We identified a subset of reviews that were extremely short (e.g., "Good", "Ok", or garbage characters). We set a threshold of 5 characters; any review shorter than this was removed, as it provides insufficient context for a machine learning model to learn meaningful patterns.

5 Feature Engineering: Sentiment Labeling

The raw dataset used a 5-star rating scale. For the purpose of sentiment analysis, a regression approach (predicting 1-5) is often less useful than a classification approach (Positive/Neutral/Negative). We engineered a new target variable, sentiment, based on the following mapping:

Star Rating	Sentiment Label	Rationale
1 Star	Negative	Strong dissatisfaction
2 Stars	Negative	Dissatisfaction
3 Stars	Neutral	Mixed or average experience
4 Stars	Positive	Satisfaction
5 Stars	Positive	Strong satisfaction

Table 1: Mapping of Star Ratings to Sentiment Classes

This transformation simplified our problem into a 3-class classification task, which is standard for sentiment analysis applications.

6 Addressing Class Imbalance

One of the most critical challenges identified during our Exploratory Data Analysis (EDA) was severe class imbalance.

6.1 The Imbalance Problem

In the tourism industry, customers are statistically more likely to leave reviews when they are satisfied. Consequently, our dataset was heavily skewed:

- **Positive Class**: Dominant majority (approx. 70-80% of data).
- **Negative Class**: Minority.
- **Neutral Class**: Severe minority.

Training a model on this skewed data would result in a "lazy" classifier that achieves high accuracy simply by predicting "Positive" for every input, while failing to detect negative feedback—which is often the most critical for businesses to address.

6.2 Synonym-Based Upsampling Strategy

To resolve this, we rejected simple random oversampling (duplicating rows), which leads to overfitting. Instead, we adopted a sophisticated **Data Augmentation** approach using the nlpaug library.

6.2.1 Methodology

We utilized the `SynonymAug` augmenter backed by the WordNet database.

- **Target**: We isolated the minority classes (Negative and Neutral).
- **Mechanism**: The algorithm iterates through the reviews and replaces a percentage of words with their synonyms (e.g., changing "The hotel was bad" to "The hotel was poor").
- **Configuration**: We set `aug_p=0.1`, meaning roughly 10% of words in a sentence were substituted. This threshold was chosen to introduce variation without altering the fundamental semantic meaning of the review.

6.2.2 Outcome

We generated synthetic samples for the Negative and Neutral classes until their counts matched the Positive class.

- **Before**: Highly skewed distribution.
- **After**: Perfectly balanced distribution (1:1:1 ratio).

This ensures that the model treats all sentiment classes with equal importance during the training phase.

7 Text Preprocessing and Vectorization

The final stage of the workflow was to convert the natural language text into a numerical format interpretable by machine learning algorithms. We applied specific preprocessing pipelines tailored to each model architecture.

7.1 Standard NLP Pipeline

We defined a `clean_text` function applied to every review:

1. **Lowercasing**: "Hotel" and "hotel" are treated as the same word.
2. **URL Removal**: Removing "http://" links which are irrelevant to sentiment.
3. **Punctuation and Number Removal**: Removing non-alphabetic characters to focus purely on words.
4. **Whitespace Normalization**: Collapsing multiple spaces into single spaces.

7.2 Baseline Vectorization: TF-IDF

For the baseline Logistic Regression model, we selected **Term Frequency-Inverse Document Frequency (TF-IDF)** as our feature extraction method. Unlike simple Bag-of-Words, TF-IDF weighs terms based on their importance:

$$TF(t,d) = \frac{count of t in d}{total words in d}$$

$$IDF(t) = \log\left(\frac{total documents}{documents containing t}\right)$$

This penalizes common words (like "the", "is") and highlights rare, descriptive words (like "filthy", "exquisite").

7.2.1 Configuration Details

- **Max Features**: 10,000. We restricted the vocabulary to the top 10,000 most frequent words to control dimensionality and prevent overfitting.
- **N-grams**: (1, 2). We included both unigrams (single words) and bigrams (pairs of words). This is crucial for capturing negations (e.g., "not good"), which would be lost in a unigram-only model.
- **Stop Words**: We removed standard English stop words to further reduce noise.

7.3 Tokenization and Embedding (Second Model)

For the advanced deep learning model (Second Model), we transitioned from the sparse TF-IDF approach to a learnable embedding architecture, which captures semantic relationships between words more effectively.

7.3.1 Tokenization

We utilized the TensorFlow `TextVectorization` layer to map raw strings to integer sequences.

- **Vocabulary Size**: We set a maximum vocabulary size of 60,000 tokens. This covers the vast majority of the corpus while pruning extremely rare words.
- **Sequence Length**: All reviews were standardized to a fixed length of 200 tokens. Reviews shorter than this were padded with zeros, while longer reviews were truncated. This uniformity is essential for batch processing in neural networks.

7.3.2 Word Embedding

Instead of sparse vectors, we employed a dense **Embedding Layer** as the first layer of our neural network.

- **Dimensionality**: Each token is represented by a dense vector of size 128.
- **Learning**: These embeddings are initialized randomly and learned jointly with the model during training. This allows the model to learn domain-specific semantic representations tailored to tourism reviews (e.g., learning that ”dirty” and ”filthy” are semantically close).

8 Conclusion

The preprocessing workflow described in this report represents a comprehensive effort to curate a high-quality dataset for tourism sentiment analysis. By critically evaluating the five original Yelp datasets, we streamlined our focus to the most relevant Review and Business data. We rigorously cleaned the text, engineered meaningful sentiment labels, and—most importantly—implemented an advanced synonym-based upsampling strategy to neutralize class imbalance. The resulting dataset is balanced, clean, and numerically vectorized, providing a solid foundation for training high-performance machine learning models.