Analysis and Nutritional Metadata Classification on Open Food Facts Dataset

Aniket Gurav

May 21, 2025

Contents

1 Introduction

This project focuses on developing a machine learning pipeline for food product classification using real-world data from the Open Food Facts database. The dataset contains information about packaged food items including their names, ingredient lists, nutritional values, and standardized category labels.

A typical example from the dataset is the product *Chocolat noir 85% cacao – J.D. Gross.*[1] This entry includes a product name, detailed ingredients, NutriScore, NOVA group, and high-level food categories such as `pnns_groups_1` and `pnns_groups_2`.

The choice of this dataset is motivated by its open access, real-world relevance, and rich combination of structured and unstructured information. These properties make it an ideal source for exploring natural language processing, feature engineering, and deep learning methods in a nutrition-focused setting.

2 Missing Value Analysis

2.1 Dataset Overview

The dataset used in this analysis is derived from the Open Food Facts project, downloaded as a gzipped TSV file (`en.openfoodfacts.org.products.csv.gz`). The raw dataset contains **3,833,367 records** and **209 columns** representing various product attributes, including nutritional values, categories, brands, and ingredients.

A cleaning process was applied to remove columns with more than 90% missing data. The filtered output was saved as `cleaned_openfoodfacts.csv` and used for all further modeling and analysis tasks.

2.2 Column Inventory

Each record in the cleaned dataset contains over 73 columns after cleaning, with data types ranging from strings and integers to floating-point values. Key columns include:

- `product_name` – Name of the food product.
- `ingredients_text` – Unstructured text listing the ingredients.
- `pnns_groups_1`, `pnns_groups_2` – High-level and sub-level food category labels.
- `energy_100g`, `protein_100g`, etc. – Nutritional information per 100g.

2.3 Missing Value Summary

To assess data completeness, a column-wise missing value analysis was performed. The percentage of missing values and data types were computed for each column. The table below summarizes the top features with missing data:

[1] https://world.openfoodfacts.org/product/20995553/chocolat-noir-85-cacao-j-d-gross

Column	Missing %	Data Type
ingredients_text	70.65%	string
pnns_groups_1	0.57%	string
pnns_groups_2	0.57%	string
product_name	6.49%	string
energy_100g	25.88%	float
fat_100g	26.68%	float
proteins_100g	26.55%	float
fiber_100g	64.31%	float
image_url	24.90%	string
nova_group	73.92%	float
vitamin-c_100g	94.09%	float
potassium_100g	94.91%	float
zinc_100g	99.34%	float
omega-3-fat_100g	99.62%	float

2.4 Insights

From the missing value report, we observe that:

- Core metadata like `product_name` and `pnns_groups_1/2` are relatively complete.
- Key nutritional columns have moderate missingness (25–30%).
- Many micronutrients (e.g., vitamins, minerals) are sparsely populated (over 90% missing).
- `ingredients_text`, though vital for our feature extraction, has roughly 70% missing data, prompting the need for row-level filtering.

These insights help justify the choice of focusing on ingredients-based modeling and motivate the need for feature engineering to address sparse nutrition data.

3 Exploratory Analysis of Product Metadata

To better understand the structure and distribution of product metadata, we performed value count analysis on four key columns: `categories`, `brands`, `countries`, and `main_category`. These fields contain human-readable descriptions that help contextualize each product's origin, branding, and classification.

3.1 Top Occurring Values

The top 10 values in each of the four columns were extracted and visualized in a combined grid (Figure **??**). These distributions offer insight into:

- The most frequent product categories and subcategories.
- Brand concentration — whether a few brands dominate the dataset.
- Geographical spread of the product origins.

4 Visual Exploration of Product Metadata and Label Distributions

4.1 Metadata Distributions

Figure 1 shows the top 10 most frequent values for four key metadata columns: `categories`, `brands`, `countries`, and `main_category`. These fields describe product classification, origin, and branding at a higher level.

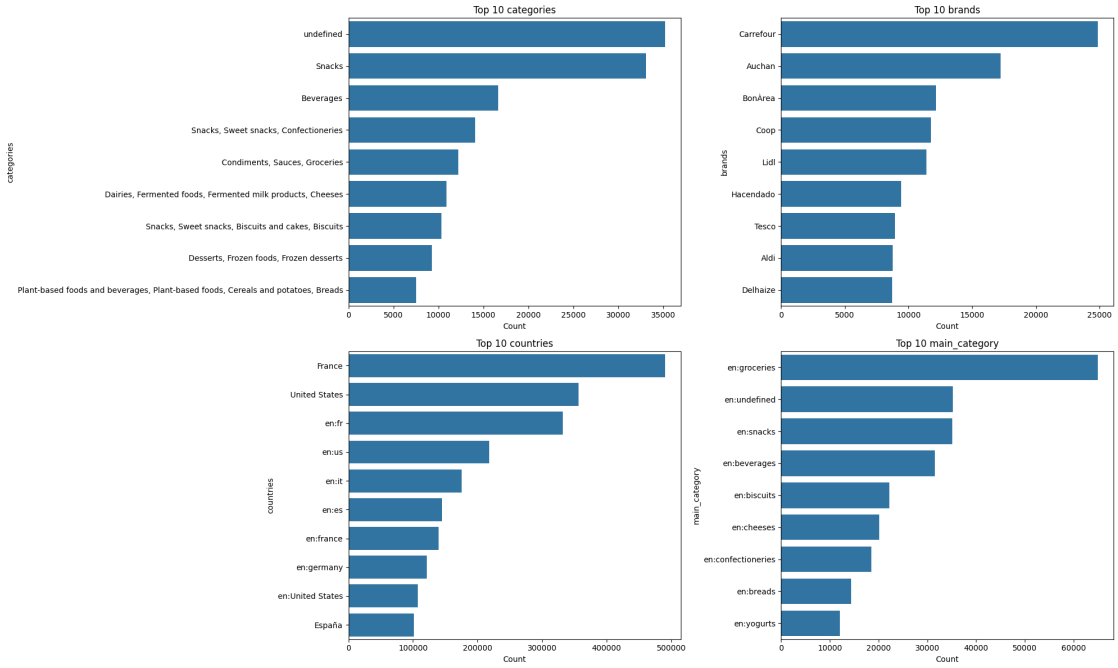


Figure 1: Top 10 value distributions for key metadata columns.

Insights:

- Certain brands and countries dominate the dataset, indicating a potential bias toward well-known companies or markets.
- `categories` and `main_category` fields are wide and varied but often redundant, which limits their value in direct classification tasks.
- These fields are useful for exploratory purposes, but inconsistent formats and missing values make them less suited for supervised learning without heavy preprocessing.

4.2 Class Distribution in Labels

Figure 2 presents the class frequency distribution for the two prediction targets: `pnns_groups_1` (broad category) and `pnns_groups_2` (fine-grained subcategory).

Insights:

- The label distributions are imbalanced — a few categories dominate while others are under-represented.

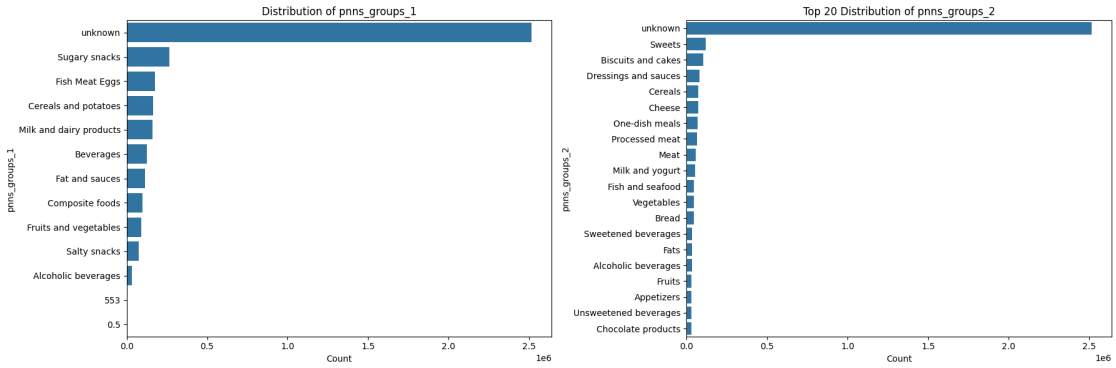


Figure 2: Class distribution of `pnns_groups_1` and top 20 of `pnns_groups_2`.

- This imbalance increases the classification challenge and motivates the use of weighted losses or stratified sampling during training.
- `pnns_groups_1` shows a manageable number of classes, suitable for high-level classification.
- `pnns_groups_2`, while more detailed, introduces more variability and requires stronger semantic signal from features (like ingredients).

These visualizations help bridge the gap between exploratory analysis and downstream modeling decisions.

4.3 Connection to Modeling Strategy

These insights help reinforce our choice of focusing on `ingredients_text` and `pnns_groups_1/2`. Although metadata columns like `brands` and `countries` are informative, they are often categorical, sparse, or inconsistent in granularity. In contrast, `ingredients_text` provides a standardized text feature from which multiple semantic properties (e.g., nutrients, additives, flavors) can be extracted and modeled.

5 Nutritional and Health Score Analysis

To complement the ingredient-based feature extraction, we explored structured nutritional and health-related attributes available in the dataset: `energy_100g`, `nova_group`, and `nutriscore_grade`. These variables provide quantitative insights into the nutritional profile, processing level, and overall healthiness of food products.

Insights

- **Energy Distribution:** The majority of food products have energy values concentrated below 1000 kcal per 100g. However, a long tail with higher values suggests the presence of highly caloric and possibly processed foods.
- **Energy by Category:** The boxplot reveals wide variation in `energy_100g` across `pnns_groups_1`. As expected, categories such as `Fat and sauces`, `Sugary snacks`, and `Composite foods` exhibit higher energy density, while `Fruits and vegetables` remain on the lower end.

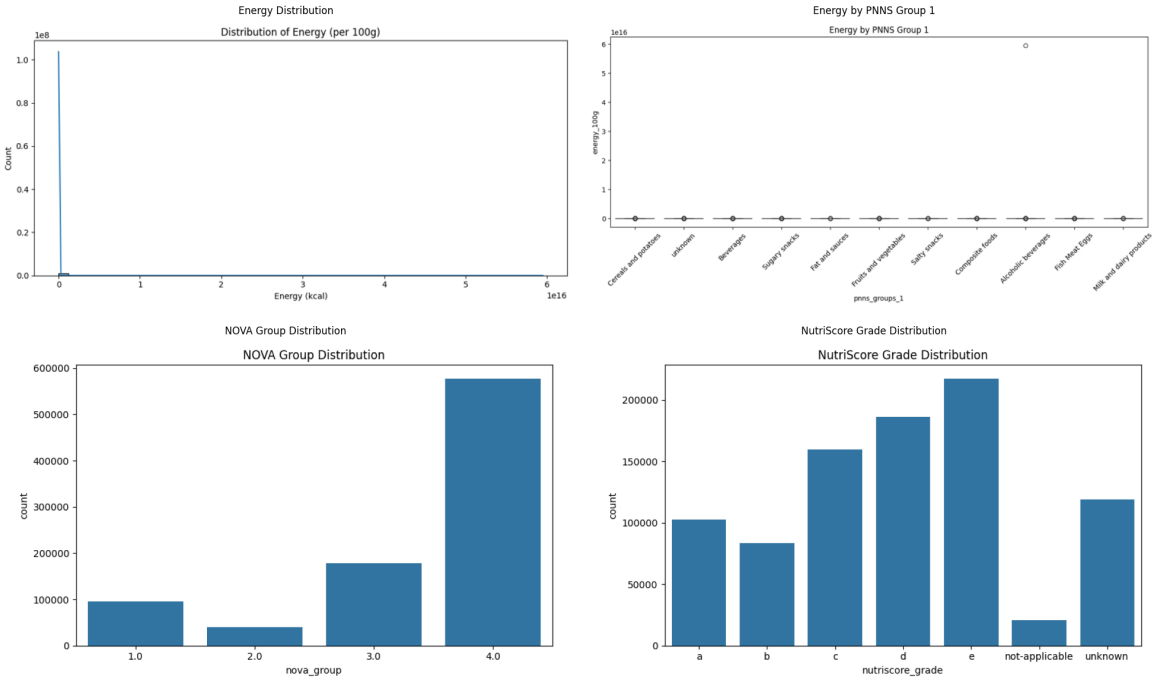


Figure 3: Combined visualization: (Top Left) Distribution of energy per 100g, (Top Right) Energy variation by food category, (Bottom Left) NOVA processing group distribution, (Bottom Right) NutriScore grade distribution.

- **NOVA Group Distribution and Mean Energy:** The NOVA classification distinguishes foods by processing level:
  - NOVA 1 (Unprocessed): 912.1 kcal/100g
  - NOVA 2 (Processed ingredients): 2573.8 kcal/100g
  - NOVA 3 (Processed foods): 1343.7 kcal/100g
  - NOVA 4 (Ultra-processed): 1.03×10^{11} kcal/100g (indicative of extreme outliers or errors)

  These values show a clear rise in energy with processing, reinforcing the importance of food processing as a predictive and health-related feature.

- **NutriScore Grade vs. Food Group:** Cross-tabulation between `nutriscore_grade` and `pnns_groups_1` shows that:
  - Grades A and B are prevalent in `Fruits and vegetables` and `Milk and dairy products`.
  - Grades D and E dominate in `Sugary snacks`, `Fat and sauces`, and `Salty snacks`.
  - `Beverages` and `Composite foods` are more evenly distributed across the score range.

  This alignment validates the classification structure and suggests that health labels could serve as evaluation baselines.

These insights reinforce the relationship between the input features (`ingredients_text`, tag counts) and nutritional outcomes, thus supporting the supervised classification approach used in this study.

[11pt]article [utf8]inputenc geometry a4paper, margin=1in graphicx booktabs enumitem listings xcolor

6 Data Preprocessing for Classification and Tagging Tasks

This section outlines the preprocessing pipeline for four tasks: (1) rule-based token tagging, (2) food category classification, (3) nutritional entity tagging, and (4) nutritional attribute prediction. Each task processes the Open Food Facts dataset to extract specific features from the `ingredients_text` and related fields, ensuring compatibility with downstream machine learning models. All tasks share a common initial filtering step, followed by task-specific processing.

6.1 Common Dataset Filtering

All tasks begin with the Open Food Facts dataset, containing over 3.8 million records. We filter it to 100,000 English-language (United States) food products with non-missing labels, retaining key fields:

- `product_name`: Product name (e.g., "Jelly Beans").
- `ingredients_text`: Ingredient list (e.g., "sugar, glucose syrup, gelatin").
- `pnns_groups_1`, `pnns_groups_2`: Broad and fine-grained category labels (e.g., "Sugary snacks", "Candies").
- `countries_tags`: Country of origin (e.g., "en:united-states").
- Nutritional fields (e.g., `energy_100g`, `proteins_100g`, `nutriscore_grade`).

Example Row:

Product: "Jelly Beans", Ingredients: "sugar, glucose syrup, gelatin", Labels: "Sugary snacks", "Candies", Country: "en:united-states".

6.2 Rule-Based Token Tagging

Task: Assign semantic tags (e.g., nutrient, flavor, unit) to each token in `ingredients_text` to create structured features.

Steps:

- Clean `ingredients_text`: Lowercase, remove special characters (e.g., "*", "&").
- Tokenize: Split into words using regex (`\b\w+\b`).
- Tag tokens: Match against curated lexicons (e.g., "protein" $\rightarrow$ NUT, "mg" $\rightarrow$ UNI, "chicken" $\rightarrow$ FLA) or quantity patterns (e.g., "10" $\rightarrow$ QTY). Unmatched tokens are tagged as ING (ingredient).
- Generate outputs: Token lists, tag lists, word-tag pairs, and tag counts (e.g., `num_NUT`, `num_FLA`).

Tags Used: Tokens are assigned one of 15 semantic tags based on nutritional and functional roles, derived from sources like USDA FoodData Central and FDA guidelines. The tags are:

Expected Data: Tokens (e.g., ["sugar", "gelatin"]), tags (e.g., ["SWE", "PRO"]), word-tag pairs (e.g., ["sugar:SWE", "gelatin:PRO"]), and tag counts (e.g., `num_SWE=1`, `num_PRO=1`).

Example:

Input: "sugar, gelatin, 10 mg" $\rightarrow$ Tokens: ["sugar", "gelatin", "10", "mg"] $\rightarrow$ Tags: ["SWE", "PRO", "QTY", "UNI"] $\rightarrow$ Counts: `num_SWE=1`, `num_PRO=1`, `num_QTY=1`, `num_UNI=1`.

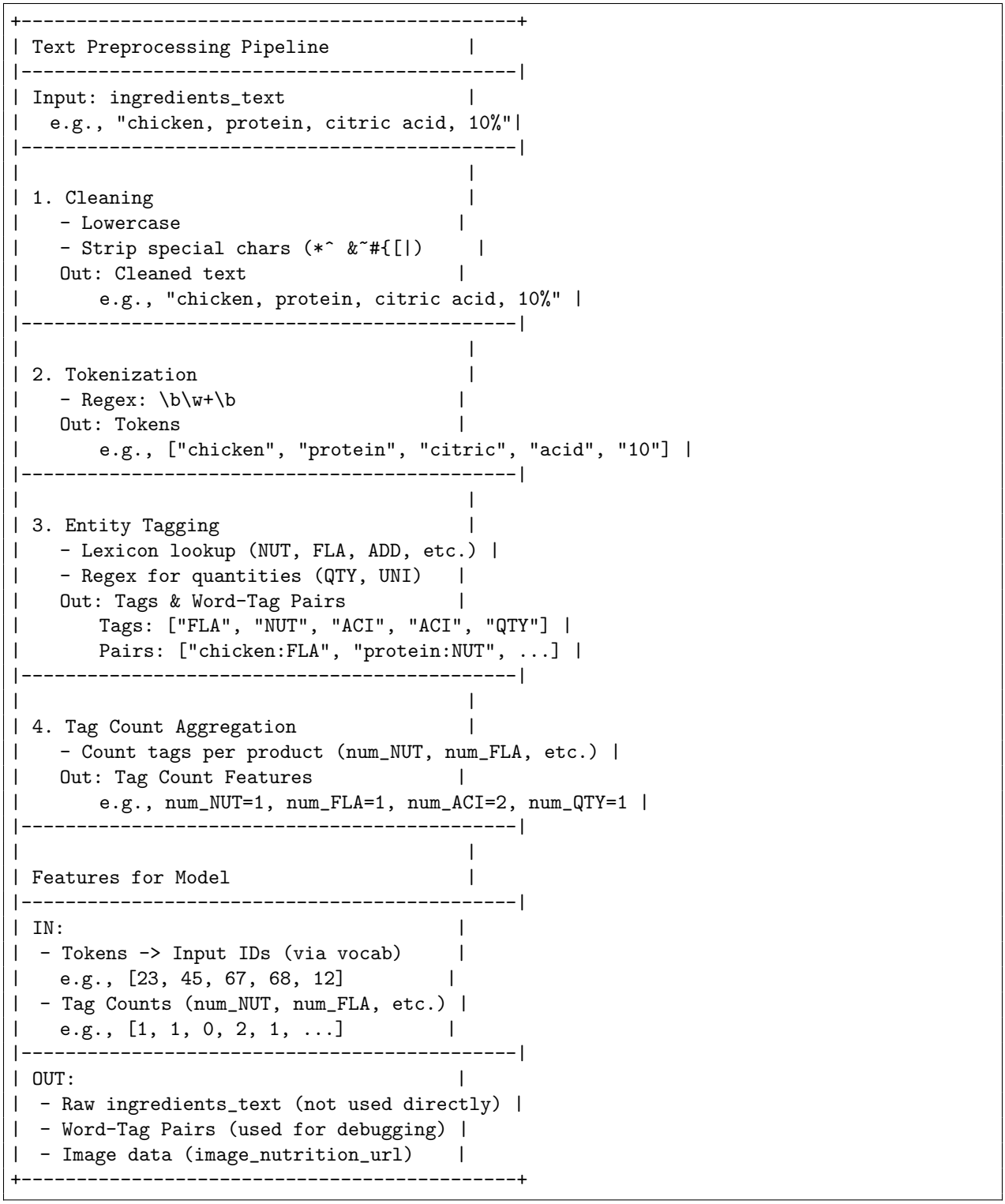


Figure 4: Text preprocessing pipeline for rule-based token tagging, illustrating input cleaning, tokenization, tagging, and feature generation for downstream tasks.

Table 2: Semantic tags for rule-based token tagging.

Tag	Description and Examples
NUT	Nutrients (e.g., protein, fiber, vitamin D)
FLA	Flavors (e.g., chicken, beef, salmon)
ADD	Additives (e.g., stabilizer, lecithin)
PRE	Preservatives (e.g., sorbic acid, sodium benzoate)
SWE	Sweeteners (e.g., sucrose, stevia)
COL	Colorants (e.g., caramel color, annatto)
MIN	Minerals (e.g., magnesium, selenium)
PRO	Probiotics (e.g., bifidobacterium, lactobacillus)
UNI	Units (e.g., mg, g, %)
THI	Thickeners (e.g., pectin, xanthan gum)
ENZ	Enzymes (e.g., lactase, amylase)
ACI	Acidifiers (e.g., citric acid, lactic acid)
HUM	Humectants (e.g., glycerol, sorbitol)
ING	Default for unclassified ingredients (e.g., rice, peas)
QTY	Quantities (e.g., 10, 0.5)

6.3 Food Category Classification

Task: Predict broad (`pnns_groups_1`) and fine-grained (`pnns_groups_2`) food categories using tokenized text and tag counts.

Steps:

- Combine `product_name` and `ingredients_text` into a unified text field.
- Clean and tokenize: Lowercase, remove punctuation, split into tokens.
- Build vocabulary: Assign indices to tokens, adding `<pad>` and `<unk>`.
- Convert tokens to indices: Create `input_ids` for model input.
- Encode labels: Convert `pnns_groups_1` and `pnns_groups_2` to numerical IDs.
- Incorporate tag counts: Use `num_NUT`, `num_FLA`, etc., from rule-based tagging as additional features.

Expected Data: Token indices (e.g., `input_ids=[12, 45, 67, 87]`), label IDs (e.g., `label_G1=3`, `label_G2=15`), and tag count features (e.g., `[num_SWE=1, num_PRO=1, ...]`).

Example:

Input: "Jelly Beans, sugar, gelatin" → Tokens: ["jelly", "beans", "sugar", "gelatin"] → `input_ids=[23, 24, 12, 87]` → Labels: `label_G1=''Sugary snacks'' (3)`, `label_G2=''Candies'' (15)` → Features: `[num_SWE=1, num_PRO=1, num_QTY=0, ...]`.

6.4 Nutritional Entity Tagging

Task: Assign BIO tags (e.g., B-NUT, I-QTY, O) to tokens in `ingredients_text` to identify nutritional entities like "Protein: 10 g".

Steps:

- Load tagged data: Use tokens and tags from rule-based token tagging output.
- Convert to BIO format: Assign B- (beginning), I- (inside), or O (outside) prefixes (e.g., "sugar cane" → ["B-SWE", "I-SWE"]).

- Build label vocabulary: Assign indices to BIO tags (e.g., `B-NUT=0`, `O=1`).
- Generate inputs: Convert tokens to indices (`input_ids`) and tags to label IDs.
- Include tag counts: Use `num_NUT`, `num_FLA`, etc., as contextual features.

Expected Data: Token indices (e.g., `input_ids=[12, 87, 10, 5]`), BIO label IDs (e.g., [0, 1, 2, 3] for ["B-SWE", "O", "B-QTY", "B-UNI"]), and tag counts.

Example:

> Input: "sugar, gelatin, 10 mg" → Tokens: ["sugar", "gelatin", "10", "mg"] → BIO Tags: ["B-SWE", "O", "B-QTY", "B-UNI"] → `input_ids=[12, 87, 10, 5]`, Label IDs: [0, 1, 2, 3] → Features: `[num_SWE=1, num_PRO=1, num_QTY=1, num_UNI=1]`.

6.5 Nutritional Attribute Prediction

Task: Predict numerical (e.g., `energy_100g`, `proteins_100g`) and categorical (e.g., `nutriscore_grade`, `pnns_groups_1`) attributes.

Steps:

- Combine fields: Merge `product_name`, `ingredients_text`, and `categories` into a unified text field.
- Clean and tokenize: Lowercase, remove punctuation, split into tokens.
- Normalize numerical targets: Clip outliers (±3 std), impute NaNs with median, scale using StandardScaler.
- Encode categorical targets: Convert `nutriscore_grade`, `pnns_groups_1` to numerical IDs.
- Encode features: Normalize `additives_n`, `serving_quantity`; encode `brands`, `countries_tags`.
- Incorporate tag counts: Use `num_NUT`, `num_FLA`, etc., from rule-based tagging.
- Build vocabulary: Convert tokens to indices (`input_ids`).

Expected Data: Token indices (e.g., `input_ids=[23, 24, 12, 87]`), normalized numerical targets (e.g., `energy_100g=0.5`), categorical label IDs (e.g., `nutriscore_grade=2`), and features (e.g., `[additives_n=0.3, serving_quantity=1.2, brands_id=45, num_SWE=1, ...]`).

Example:

> Input: "Jelly Beans, sugar, gelatin, categories: candies" → Tokens: ["jelly", "beans", "sugar", "gelatin", "candies"] → `input_ids=[23, 24, 12, 87, 90]` → Targets: `energy_100g=0.5`, `proteins_100g=0.1`, `nutriscore_grade=''D'' (3)` → Features: `[additives_n=0.2, serving_quantity=0.8, brands_id=45, num_SWE=1, num_PRO=1, ...]`.

7 Food Category Classification

Task and Motivation: Predict broad (`pnns_groups_1`, e.g., "Sugary snacks") and fine-grained (`pnns_groups_2`, e.g., "Candies") food categories to enable automated nutritional categorization. This task leverages `ingredients_text` and `product_name` to capture semantic patterns, addressing the assignment's goal of classifying food products using text features.

Steps:

- *Text Consolidation*: Concatenate `product_name` and `ingredients_text` into a unified field.
- *Cleaning and Tokenization*: Lowercase, remove punctuation, and split into tokens using regex (""b"w+"b").

- *Vocabulary Construction*: Assign indices to tokens (minimum frequency 5), adding `<pad>` and `<unk>` for padding and unknown words.
- *Token Indexing*: Convert tokens to `input_ids` (e.g., `[23, 24, 12, 87]`).
- *Label Encoding*: Map `pnns_groups_1` and `pnns_groups_2` to numerical IDs using `LabelEncoder`.
- *Feature Augmentation*: Incorporate tag counts (e.g., `num_NUT`, `num_SWE`) from rule-based tagging as contextual features.

Expected Data: Token indices (`input_ids=[12, 45, 67, 87]`), label IDs (`label_G1=3`, `label_G2=15`), and tag counts (`[num_SWE=1, num_PRO=1, ...]`).

Example:

> Input: "Jelly Beans, sugar, gelatin" → Tokens: ["jelly", "beans", "sugar", "gelatin"] → `input_ids=[23, 24, 12, 87]` → Labels: `label_G1=''Sugary snacks'' (3)`, `label_G2=''Candies'' (15)` → Features: `[num_SWE=1, num_PRO=1, num_QTY=0, ...]`.

Model Architecture: The `LSTMG1` model in `LSTM2.py` processes inputs with a GloVe embedding layer (300D, trainable), followed by a 3-layer bidirectional LSTM (50 hidden units, dropout 0.3). Tag counts are transformed via a linear layer (14 to 50 units, ReLU), concatenated with LSTM outputs, and passed through a final linear layer to predict category probabilities.

Impact: This pipeline integrates semantic tags with text features, enhancing LSTM performance for imbalanced category prediction, aligning with the assignment's classification objectives.

8 Training Procedure

Setup: The LSTM model for Food Category Classification is trained using PyTorch with GPU acceleration. We use cross-entropy loss for multi-class prediction of `pnns_groups_1` and `pnns_groups_2`. The Adam optimizer (learning rate 0.001) optimizes the model, with a batch size of 32 over 10 epochs.

Training Details: Variable-length sequences are padded to the maximum length per batch using a custom batch preparation method. Dropout (0.3) and batch normalization reduce overfitting. Model weights are selected based on validation accuracy, with evaluation on a held-out test set.

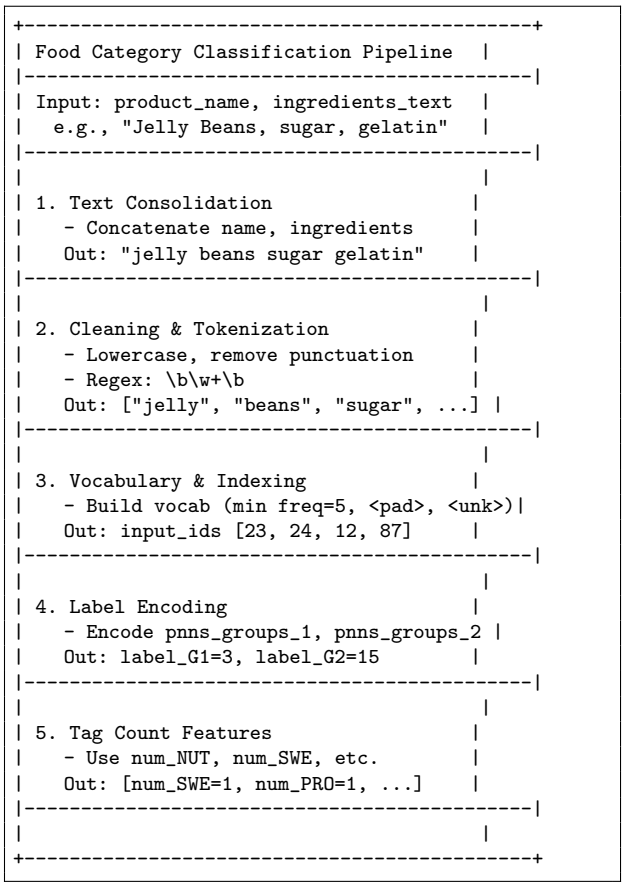
Impact: This setup ensures robust training, handling class imbalance and sequence variability, aligning with the assignment's classification goals.

8.1 Nutritional Entity Tagging

Task and Motivation: Assign BIO tags (e.g., B-NUT, I-QTY, O) to tokens in `ingredients_text` to extract nutritional entities like "Protein: 10 g". This task fulfills the assignment's entity tagging requirement, enabling automated nutritional analysis for dietary applications.

Steps:

- *Load Tagged Data*: Use tokens and tags (e.g., NUT, QTY) from rule-based tagging output.
- *BIO Conversion*: Assign B- (beginning), I- (inside), or O (outside) prefixes (e.g., "sugar cane" → ["B-SWE", "I-SWE"]).
- *Label Vocabulary*: Map BIO tags to indices (e.g., `B-NUT=0`, `O=1`).
- *Input Generation*: Convert tokens to `input_ids` and tags to `bio_label_ids`.
- *Feature Integration*: Include tag counts (e.g., `num_NUT`, `num_FLA`) as contextual features.



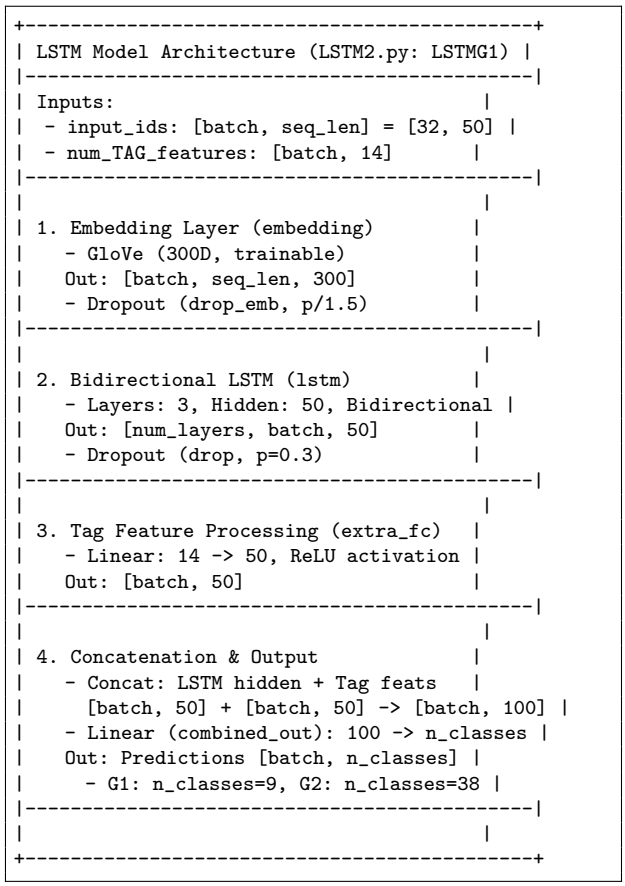


Figure 5: Classification pipeline (left) and LSTM architecture (right) for Food Category Classification, illustrating data flow and model structure.

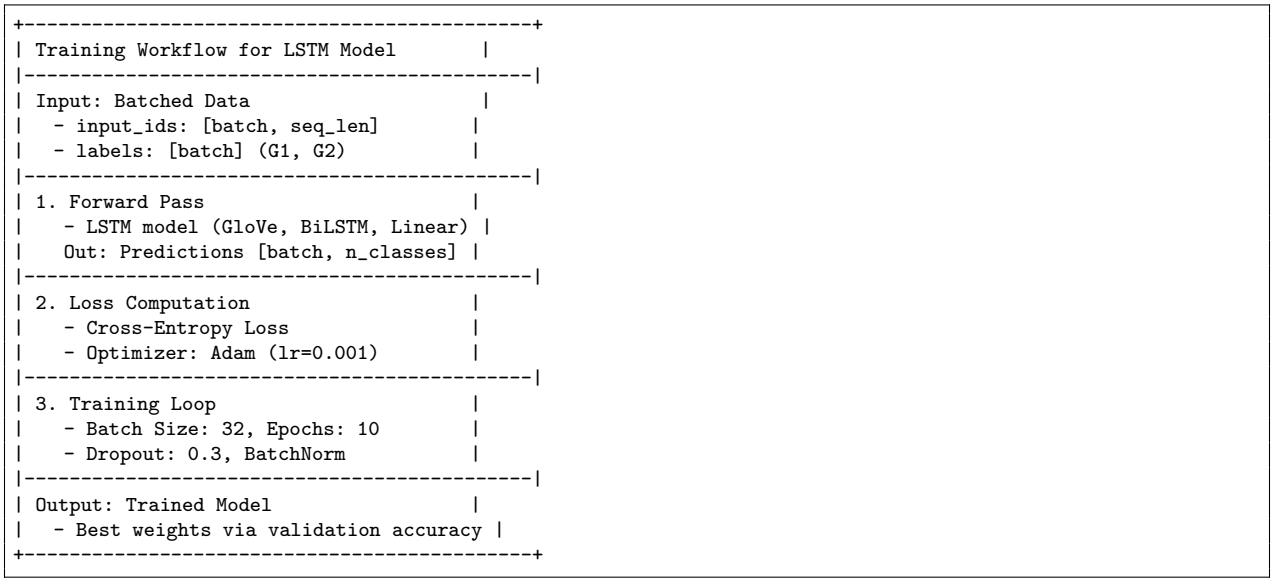


Figure 6: Training workflow for Food Category Classification using the LSTM model.

Expected Data: Token indices (`input_ids=[12, 87, 10, 5]`), BIO label IDs ([0, 1, 2, 3] for ["B-SWE", "O", "B-QTY", "B-UNI"]), and tag counts.

Example:

> Input: "sugar, gelatin, 10 mg" → Tokens: ["sugar", "gelatin", "10", "mg"] → BIO Tags: ["B-SWE", "O", "B-QTY", "B-UNI"] → `input_ids=[12, 87, 10, 5]`, Label IDs: [0, 1, 2, 3] → Features: `[num_SWE=1, num_PRO=1, num_QTY=1, num_UNI=1]`.

Impact: This pipeline extracts structured nutritional entities, supporting health applications and fulfilling assignment objectives.

8.2 Nutritional Attribute Prediction

Task and Motivation: Predict numerical attributes like `energy_100g` and `proteins_100g`, and categorical attributes like `nutriscore_grade` and `pnns_groups_1`, to enable comprehensive nutritional profiling. This task supports health-aware applications by leveraging text and structured features.

Steps:

- *Field Combination*: Merge `product_name`, `ingredients_text`, and `categories` into a unified text field.
- *Cleaning and Tokenization*: Lowercase, remove punctuation, and split into tokens.
- *Numerical Target Normalization*: Clip outliers (±3 standard deviations), impute missing values with the median, and scale using a standard scaler.
- *Categorical Target Encoding*: Convert `nutriscore_grade` and `pnns_groups_1` to numerical IDs.
- *Feature Encoding*: Normalize `additives_n` and `serving_quantity`; encode `brands` and `countries_tags`.
- *Tag Count Integration*: Use counts like `num_NUT` and `num_FLA` from rule-based tagging.
- *Vocabulary Building*: Convert tokens to indices (`input_ids`).

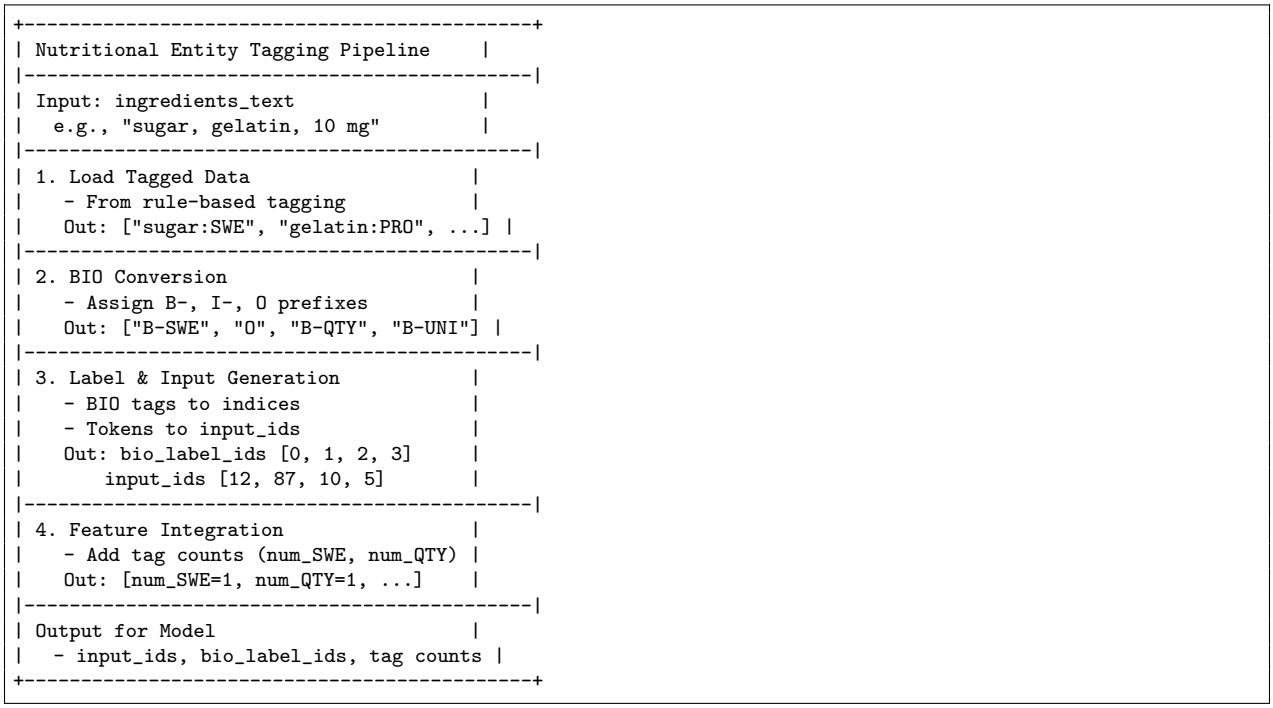


Figure 7: Pipeline for Nutritional Entity Tagging, showing BIO tag assignment and data preparation.

Expected Data: Token indices (`input_ids=[23, 24, 12, 87]`), normalized numerical targets (`energy_100g=0.5`), categorical label IDs (`nutriscore_grade=2`), and features (`[additives_n=0.3, serving_quantity=1.2, brands_id=45, num_SWE=1, ...]`).

Example:

> Input: "Jelly Beans, sugar, gelatin, categories: candies" → Tokens: ["jelly", "beans", "sugar", "gelatin", "candies"] → `input_ids=[23, 24, 12, 87, 90]` → Targets: `energy_100g=0.5`, `proteins_100g=0.1`, `nutriscore_grade=''D'' (3)` → Features: `[additives_n=0.2, serving_quantity=0.8, brands_id=45, num_SWE=1, num_PRO=1, ...]`.

9 Evaluation Metrics

Overview: We assess model performance across three tasks using task-specific metrics, reflecting classification, tagging, and regression objectives.

Food Category Classification: For predicting `pnns_groups_1` and `pnns_groups_2`, we use overall accuracy and macro/weighted averages of precision, recall, and F1-score. These metrics account for class imbalance, ensuring fair evaluation across categories. Accuracy measures the proportion of correct predictions, while F1-score balances precision and recall for underrepresented classes.

Nutritional Entity Tagging: For BIO tag prediction, we compute per-token precision, recall, and F1-score, focusing on entity-level correctness (e.g., correctly identifying "Protein: 10 g"). These metrics evaluate sequence labeling accuracy, ignoring padded tokens.

Nutritional Attribute Prediction: For numerical predictions (e.g., `energy_100g`), we use Mean Absolute Error (MAE) and R-squared (R^2). MAE measures prediction error magnitude,

while R^2 assesses the proportion of variance explained. For categorical predictions (e.g., `nutriscore_grade`), we apply the same classification metrics as above.

Rationale: These metrics align with the assignment's focus on classification and regression quality, providing a comprehensive performance evaluation across diverse tasks.

Conclusion and Future Work ¡xaiArtifact artifact$_i d$ = "35bb58a9−4cd1−4374−b60c−3f9507ab666a"$artifact_v er$ "28e1c9b2−63d8−42e3−9e15−b031e827a09a"$title$ = "$ConclusionandFutureWork$"$contentType$ = "$text/latex$" >

10 Conclusion and Future Work

Summary: This project developed a pipeline for food product analysis using the Open Food Facts dataset, covering rule-based token tagging, food category classification, nutritional entity tagging, and attribute prediction. The LSTM model, enhanced with GloVe embeddings and tag counts, effectively predicted categories (`pnns_groups_1/2`), entities (e.g., "Protein: 10 g"), and nutritional values, demonstrating the feasibility of text-based nutritional analysis.

Key Findings: The rule-based tagging system provided structured features, improving model performance. Classification and tagging tasks handled imbalanced data well, while attribute prediction offered insights into nutritional profiling, supporting health applications.

Future Work:

- Incorporate image data for multimodal learning to enhance entity and attribute prediction.
- Use structured nutrient fields as additional inputs for improved accuracy.
- Address class imbalance with weighted loss or data augmentation techniques.
- Explore transformer-based models for better sequence modeling and scalability.

This pipeline establishes a strong foundation for automated nutritional analysis, with potential for broader health-aware applications.