Detailed Explanation of the Normalization Approach

Objective

The task required developing a robust and potentially scalable solution for text normalization in the music publishing domain. The aim was to process raw textual data containing composer and writer information, remove irrelevant or redundant information, and retain only meaningful names while handling multilingual data. Given the noisy and inconsistent nature of real-world datasets, the solution needed to be generalizable to unseen data and demonstrate proficiency in handling ambiguous cases.

Overview of the Approach

Our normalization pipeline combines rule-based techniques, regex-based preprocessing, and Named Entity Recognition (NER) to achieve accurate text normalization. This multi-step approach is designed to clean, validate, and reorder names while excluding placeholders, publishers, and other irrelevant data. Below is a detailed breakdown of each component:

1. Preprocessing and Cleaning

Purpose

The raw input text often contained:

- **Delimiters**: Multiple inconsistent delimiters such as /, ,, and &.
- **Placeholders**: Keywords like "Unknown Writer" or "Copyright Control" that needed to be excluded.
- **Irrelevant Parentheses**: Phrases like "(PERF BY ...)" or "(performed by ...)" that added no value to the final output.
- **Numeric Tokens**: Strings like "(999990)" that acted as placeholders for missing data.
- **Multilingual and Non-Latin Characters**: Names in various scripts (e.g., Cyrillic, Chinese, or accented Latin characters).

Techniques

1. **Unifying Delimiters**:
    - Replaced inconsistent delimiters (e.g., ,, &) with / for consistency. This ensured that splitting and processing the text segments was streamlined.
2. **Exclusion Rules**:

- A predefined dictionary (keywords_to_exclude) was used to identify and exclude irrelevant tokens such as "Unknown Writer," "Public Domain," or publisher names like "Sony/ATV Music Publishing."
- Regex patterns (regex_patterns) were applied to remove numeric placeholders or patterns like "(PERF BY ...)."

3. **Regex-Based Parentheses Removal**:
   - A custom function, remove_perf_by_phrases, was implemented to specifically handle patterns like "(PERF BY ...)" and "(performed by ...)," which frequently appeared in the dataset.
4. **Unicode Handling**:
   - The solution ensured that Unicode characters were preserved during processing, enabling correct handling of non-Latin names like "Nguyễn Nhật Huy" or "Жора Иванов."

Outcome

This step transformed raw input into clean, segmentable strings that were ready for deeper validation and normalization.

2. Segment Validation and Entity Recognition

Purpose

After cleaning, each segment was analyzed to determine if it represented a valid composer or writer name. This was crucial for excluding irrelevant information (e.g., publishers or placeholder text) while retaining meaningful data.

Techniques

1. **NER Using spaCy**:
   - **Multilingual NER Model**: We used xx_ent_wiki_sm, a multilingual spaCy model, to identify named entities.
   - **Entity Types**:
     - PERSON: Segments labeled as PERSON were prioritized and directly included in the normalized output.
     - ORG and GPE: These were handled with caution. For example, single-word ORG or GPE entities (e.g., "Valve") were excluded, while multi-word entities were retained only if they seemed to represent meaningful names.
2. **Heuristics for Validation**:

- If spaCy failed to label a segment but the segment contained at least two alphabetic characters, it was treated as a fallback and retained.
- Exclusion rules were applied again to ensure segments containing only excluded keywords or single irrelevant words were removed.

3. **Handling Ambiguities**:
    - Multi-word segments not explicitly identified as PERSON were retained to avoid over-exclusion (e.g., "Kodak Black").
    - Single-word entities were excluded unless explicitly validated by spaCy or deemed meaningful based on contextual knowledge.

Outcome

This step significantly reduced false positives and ensured that retained segments were valid and meaningful names.

3. Name Reordering

Purpose

Many names in the dataset were presented in "Lastname, Firstname" format, which is less intuitive and inconsistent with other formats. This step ensured uniformity in the normalized output.

Techniques

- **Comma-Based Reordering**:
    - A function, name_reorder, was implemented to reorder names with commas into "Firstname Lastname" format.
    - This was applied conditionally, ensuring names without commas remained unaffected.

Outcome

All retained names were reordered into a consistent format, improving readability and consistency.

4. Normalization and Aggregation

Purpose

The cleaned, validated, and reordered segments were aggregated into a single normalized string for each row in the dataset.

Techniques

- **Slash-Separated Aggregation**:

- Retained segments were joined with slashes (/) to produce the final normalized output.
- This format aligned with the ground truth in the dataset.

Outcome

Each row in the dataset was transformed into a fully normalized, slash-separated string of valid composer/writer names.

5. Evaluation and Metrics

Purpose

To measure the effectiveness of the normalization pipeline and identify areas for improvement, we evaluated the normalized output against the ground truth.

Techniques

1. **Exact Match**:
   - Percentage of rows where the normalized output exactly matched the ground truth.
2. **Jaccard Similarity**:
   - Computed the overlap between the predicted and expected names as the ratio of intersection to union.
   - Useful for identifying partial matches and quantifying similarity in ambiguous cases.
3. **Visualization**:
   - A histogram of Jaccard similarities provided insights into the distribution of results, highlighting the proportion of rows with high, medium, and low similarity.

Outcome

The metrics demonstrated high accuracy, with a majority of rows achieving perfect or near-perfect matches. Rows with low Jaccard similarity were analyzed to identify failure cases.

6. Handling Edge Cases

Purpose

Edge cases such as ambiguous single-word names, non-Latin characters, or placeholders required special attention.

Techniques

- **Fallback Logic**:
    - Retained single-word segments as a last resort if they contained valid alphabetic characters and were not in the exclusion list.
- **Multilingual Support**:
    - Unicode characters were preserved, ensuring correct handling of names in various scripts.
- **Threshold-Based Analysis**:
    - Rows with low Jaccard similarity were flagged for manual review, enabling iterative improvements to the pipeline.

Final Deliverables

1. **Jupyter Notebook:**
    - A Jupyter Notebook containing the implementation of the normalization solution.
2. **Example Outputs** (at the end of the Jupyter Notebook):
    - A data frame sample at the bottom of the Jupyter notebook demonstrating how the solution processes the dataset.
3. **Evaluation Metrics** (inside the Jupyter Notebook):
    - Exact Match, Average Jaccard Similarity, and visualizations for comprehensive analysis.
4. **Normalized Dataset**:
    - A CSV file containing the raw text, ground truth, and normalized output.
5. **Suggestions for Improvement and Scaling**:
    - A file containing the suggestions for scaling and improving the solution.
6. **Brief report**
    - A README file included in the repository as a brief report summarizing the approach, findings, and recommendations.