Preprocessing Script 1: JSONL to BIO Format Conversion

`jsonl2bio.py`

Overview

This script converts annotations from Doccano's JSONL format into a BIO (Beginning, Inside, Outside) tagging format, suitable for training a Named Entity Recognition (NER) model using BERT. It's specifically designed for the domain of security vulnerabilities.

Features

1. **Regex-Based Error Correction:** Utilizes regular expressions to identify and correct potential human errors in labeling, ensuring more accurate entity recognition.
    a. **CVE identifiers:** To capture Common Vulnerabilities and Exposures (CVE) identifiers.
        i. Pattern: `**\bCVE-\d+-\d+\b**`
        ii. Example Match: "CVE-2021-34527"
    b. **CVSS Scores (Risk):** To find CVSS (Common Vulnerability Scoring System) scores.
        i. Pattern: `**\b\d+(\.\d+)?\/[A-Z]{2}:[A-Z]\/[A-Z]{2}:[A-Z]\/[A-Z]{2}:[A-Z]\/[A-Z]{2}:[A-Z]\/[A-Z]{1}:[A-Z]\/[A-Z]{1}:[A-Z]\/[A-Z]{1}:[A-Z](,[A-Z])?\b**`
        ii. Example Match: "9.0/AV:N/AC:L/Au:N/C:P/I:P/A:P"
    c. **CWE identifiers (Vulnerability Type):** For identifying Common Weakness Enumeration (CWE) identifiers.
        i. Pattern: `**\bCWE-\d+\b**`
        ii. Example Match: "CWE-79"
    d. **Host Inforamtion Patterns:** To detect mentions of various operating systems, which can be indicative of host information.
        i. Patterns:
            1. `**\bubuntu\b**`
            2. `**\blinux\b**`
            3. `**\bwindows\b**`
            4. `**\bmacos\b**`
            5. `**\bios\b**`
            6. `**\bandroid\b**`
        ii. Example Match: "Ubuntu"
    e. **Tested On Pattern:** Specifically for sentences that start with "Tested on" followed by a word, usually indicating an operating system.

    i. Pattern: `**Tested on\s+(\w+)**`
    ii. Example Match: "Tested on Windows"
2. 2. **Exclusion of Specific Labels:** Ignores certain entity types like ‘Proof of Concept’ and ‘Steps to Reproduce’ due to their extensive length, which may not be effectively processed by the BERT model.
3. **Entity Extraction and Label Mapping:** Extracts entities from text and maps then to predefined labels, adapting them for NER training.

Input and Output

- **Input:** JSONL file from Doccano.
- **Output:** CSV file with text and corresponding BIO labels.

Functions

1. **`map_labels(label) -> str`:** Maps Doccano labels to predefined labels used in the model.
2. **`extract_entities(text, entities) -> list`:**
    a. Splits the text into words and initializes labels to ‘O’ (Outside).
    b. Applies regular expressions to capture unlabeled entities.
    c. Labels entities according to the BIO format.
3. **`convert_to_csv(jsonl_file, csv_file, invalid_json_file)`:**
    d. Reads annotations from a JSONL file, handling any JSON decoding errors.
    e. Writes the text and corresponding BIO labels to a CSV file.
4. **`create_multi_label_csv(jsonl_file, csv_file)`:**
    f. Identifies and records instances where entities overlap with different labels, which is crucial for understanding complex annotations.

Preprocessing Script 2: Undersampling BIO-tagged Data

`undersampling.py`

Overview

This script is designed to balance the distribution of BIO tags in a dataset generated from the first preprocessing script. It specifically focuses on reducing the number of ‘O’ (Outisde) labels to match the number of ‘B’ (Beginning) labels, addressing the issue of data imbalance in NER tasks.

Features

1. **Data Balancing:** Addresses the common issue of class imbalance in NER datasets by equalizing the occurences of entity (‘B’ and ‘I’) and non-entity (‘O’) labels.

Input and Output

- **Input:** Original CSV file with BIO-tagged data.
- **Output:** New CSV file with balanced ‘B’ and ‘O’ labels.

Functions

1. **`count_b_labels(csv_file) -> int`:**
    a. Counts the total number of ‘B’ labels in the original CSV file.
    b. Returns the count for guiding the undersampling process.
2. **`convert_to_csv(jsonl_file, csv_file, invalid_json_file, original_csv_file)**
    a. Performs the undersampling process.
    b. Reads the original BIO-tagges data and identifies ‘O’ labels to be randomly removed.
    c. Writes the balanced data into a new CSV file, maintaining text integrity and adjusted BIO labels.

Preprocessing Script 3: Formatting BIO-tagged Data for NER Model

`formatting.py`

Overview

This script reformats the balanced BIO-tagged dataset into a format that aligns with the requirements of the NER model. It adds sentence numbering and placeholder POS tags, which may be necessary for specific NER model architectures.

Features

1. **Sentence Numbering:** Assigns a unique identifier to each sentence, aiding the model in contextual understanding.
2. **POS Tag Placeholder:** Adds a placeholder 'NONE' for the POS tag, required by some NER model formats.

Input and Output

- **Input:** CSV file with balanced BIO-tagged data (from Preprocessing Script 2).
- **Output:** Reformatted CSV file with sentence numbers, words, placeholder POS tags, and BIO labels.

Functions

1. **Main Script:**
    a. Reads the input CSV file containing the balanced BIO-tagged data.
    b. Writes a new CSV file with additional columns: 'Sentence #', 'Word', 'POS', 'Tag'.
    c. Increments the sentence number based on the occurrence of a full stop ('.') in the words.

Preprocessing Script 4: Oversampling Minority Tags

`oversampling.py`

Overview

This script is designed to balance a Named Entity Recognition (NER) dataset by oversampling sentences containing minority tags. It addresses class imbalance by identifying and duplicating sentences with less frequent tags, maintaining the contextual integrity of the data, which is crucial for NER tasks. This approach is particularly effective for textual data where traditional numerical oversampling methods like SMOTE may not be appropriate.

Features

1. **Dynamic Tag Count Analysis:** Calculates the occurrence of each 'B-' tag to identify minority tags.
2. **Context Preservation:** Duplicates entire sentences containing minority tags, ensuring that the oversampled data maintains contextual integrity.
3. **Adjustable Oversampling Thresholds:** Allows setting of specific thresholds and caps for oversampling, providing flexibility based on the dataset's specific needs.

Input and Output

- **Input:** CSV file with balanced and formatted BIO-tagged data (from Preprocessing Script 3).
- **Output:** CSV file with oversampled minority tags.

Functions

1. **Main Script:**
    a. Reads the formatted dataset from a CSV file.
    b. Dynamically calculates the count of each 'B-' tag in the dataset.
    c. Identifies tags that fall below a certain threshold (e.g., 80% of the maximum tag count) and classifies them as minority tags.

d. Calculates the required duplication multiplier for sentences containing each minority tag.
e. Duplicates these sentences accordingly.
f. Shuffles the duplicated sentences to avoid sequence bias.
g. Combines the original data with the oversampled sentences into a new DataFrame.
h. Shuffles the entire dataset to ensure a random distribution.
i. Saves the newly balanced dataset to a CSV file.