Design Document for NGram and WordNet Projects

Project Overview

The NGram and WordNet projects are designed to provide tools for analyzing language data, extracting word relationships, and visualizing trends over time. These projects include handling datasets such as NGram files (word usage frequency over time) and WordNet files (word relationships, including synonyms and hyponyms). Together, they allow for queries about word relationships, historical word trends, and data visualizations.

Key Components

1. **NGramMap**: Handles time-series data about word usage frequencies from NGram datasets.
2. **TimeSeries**: Represents a mapping of years to numerical data, supporting operations like summation and division.
3. **WordNet**: Manages word relationships using synset and hyponym data files.
4. **HyponymsHandler**: Processes queries for hyponyms and word relationships, supporting features like filtering by time range or limiting the result size (k).

Data Files

1. **NGram Data Files**:
    - **Words File**: Contains rows with the format <word>\t<year>\t<count>. Each row specifies how often a word was used in a specific year.
    - **Counts File**: Contains rows with the format <year>,<total_words_in_year>.
2. **WordNet Data Files**:
    - **Synsets File**: Contains rows with the format <synset_id>,<words>,<definition>. Each synset maps an ID to a list of synonyms.
    - **Hyponyms File**: Contains rows with the format <synset_id>,<hyponym_id_1>,<hyponym_id_2>,.... Each row describes hyponym relationships.

Class Descriptions

1. NGramMap

Purpose: Handles word usage frequency data and provides tools for querying and aggregating trends over time.

Important Methods:

- **NGramMap(String wordsFilename, String countsFilename)**:
    - Loads the words and counts files into data structures (wordmap for word frequencies and counts for total yearly counts).
- **countHistory(String word)**:
    - Returns a TimeSeries for the total count of a word across all years.
- **weightHistory(String word, int startYear, int endYear)**:
    - Returns a TimeSeries of the relative frequency of a word between startYear and endYear.
- **summedWeightHistory(Collection<String> words)**:
    - Returns a TimeSeries for the summed relative frequency of multiple words.

Key Data Structures:

- **wordmap**: A TreeMap<String, TimeSeries> mapping words to their time-series data.
- **counts**: A TimeSeries for total word counts per year.

---

2. TimeSeries

Purpose: Represents a mapping of years to numerical data and supports arithmetic operations.

Important Methods:

- **plus(TimeSeries ts)**:
    - Adds the values of two TimeSeries year-by-year.
- **dividedBy(TimeSeries ts)**:
    - Divides the values of one TimeSeries by another year-by-year.
- **years()**:
    - Returns a list of all years in ascending order.

Key Features:

- Extends TreeMap<Integer, Double>, leveraging its natural ordering and efficient operations.
- Designed for defensive copying and arithmetic operations.

---

3. WordNet

Purpose: Handles word relationships using synsets and hyponym data.

Important Methods:

- **WordNet(String synsetFile, String hyponymFile)**:
    - Constructs the WordNet graph using the provided data files.
- **givemehyponyms(String noun)**:
    - Returns all hyponyms (direct and indirect) for a given noun.
- **parseSyn(String synFile)**:
    - Parses the synset file to build mappings from nouns to synsets and synsets to words.
- **parsHypo(String hypoFile)**:
    - Parses the hyponym file to build the hyponym graph.

Key Data Structures:

- **idToSynset**: A Map<Integer, String[]> mapping synset IDs to word lists.
- **nounGraph**: A Graph<String, Integer> mapping nouns to their synset IDs.
- **hyponymGraph**: A Graph<Integer, Integer> mapping synset IDs to their hyponyms.

---

4. HyponymsHandler

Purpose: Processes queries for word relationships, such as finding hyponyms of a word or multiple words.

Important Methods:

- **handle(NgordnetQuery q)**:
    - Processes a query and returns results (e.g., hyponyms, filtered by time range or limited by k).
- **Intersection Logic**:
    - Finds the common hyponyms for multiple words.

Key Features:

- Handles queries efficiently using the WordNet and NGramMap objects.
- Supports filtering results by popularity over a time range (k most popular results).

Key Design Decisions

1. **Separation of Concerns**:
    - NGramMap and WordNet handle distinct datasets, ensuring modularity.
    - HyponymsHandler acts as a wrapper for these classes, enabling clean and efficient query handling.
2. **Recursive Traversal**:
    - Recursive depth-first search is used in WordNet to ensure all descendants are included in hyponym queries.
3. **Defensive Copying**:
    - Methods in TimeSeries and NGramMap ensure that modifications to returned data do not affect the underlying dataset.
4. **Sorting Logic**:
    - Results are sorted alphabetically after filtering, ensuring consistent outputs.

Challenges and Solutions

1. **Cycle Handling in WordNet**:
    - Implemented a visited set to prevent infinite loops during graph traversal.
2. **Performance with Large Data**:
    - Used efficient data structures (TreeMap, HashSet) to optimize lookups and sorting.
3. **Handling Real-World Data Issues**:
    - Handled invalid or incomplete lines in input files with error-checking logic.