Ameen Amer – 325731990

Raya Wattad - 324883388

Design Report

System Overview

The system is designed to calculate association vectors for lexemes and then compute similarity measures between these vectors. The process is divided into three main steps:

1. **Step 1: Counting Variables for Association Vectors using trivial count words.**
    - **Input:** Biarcs files containing lexeme-feature pairs.
    - **Output:**
        - Pairs of (lexeme, feature) with their counts emitted to Step 2.
        - The gold standard lexemes and top 1000 features their counts written to separate files on S3.
        - Global variables (L* and F*) are also calculated and stored.
2. **Step 2: Calculating Association Vectors using reducer side join.**
    - **Input:** Pairs of (lexeme, feature) with their counts from Step 1.
    - **Output:** Four association vectors (frequency, probability, PMI, t-test) for each lexeme, calculated based on specific equations. These vectors are emitted for further processing.
3. **Step 3: Calculating Similarities using fuzzy join**
    - **Input:** Association vectors from Step 2.
    - **Output:** A 24-dimensional vector representing the similarity measures between each pair of vectors, calculated using six different similarity metrics.

Components and Functionality

1. **Step 1: Mapper**
    - **Input:** Biarcs files.
    - **Output:** Emits lexeme/feature/pairs/global variables with their counts.
    - **Key-Value Pairs:**
        - Key: lexeme, feature or global variable identifiers (L*, F*).
        - Value: Count of the lexeme/feature/pair or global variable value.
    - **Memory Usage:** Moderate, as it processes each line of the input file and maintains a set of related lexemes.
2. **Step 1: Reducer**

- **Input:** Aggregated counts from the mapper.
- **Output:** Writes global variables and top features/lexemes to S3.
- **Key-Value Pairs:**
  - Key: (lexeme, feature) or global variable identifiers.
  - Value: Aggregated count.
- **Memory Usage:** High, as it maintains lists of top features and lexemes.

3. **Step 2: Mapper**
   - **Input:** (lexeme, feature) pairs with their counts.
   - **Output:** Emits lexemes with their corresponding feature counts.
   - **Key-Value Pairs:**
     - Key: Lexeme.
     - Value: Feature and its count.
   - **Memory Usage:** Low, as it processes each pair individually.
4. **Step 2: Reducer**
   - **Input:** Lexemes with their feature counts.
   - **Output:** Calculates and emits association vectors for each lexeme.
   - **Key-Value Pairs:**
     - Key: Lexeme.
     - Value: Association vectors (frequency, probability, PMI, t-test).
   - **Memory Usage:** Moderate, as it maintains maps of feature and lexeme counts.
5. **Step 3: Mapper**
   - **Input:** Association vectors from Step 2.
   - **Output:** Emits pairs of lexemes with their association vectors – Fuzzy join.
   - **Key-Value Pairs:**
     - Key: Pair of lexemes.
     - Value: Association vectors.
   - **Memory Usage:** Low, as it processes each vector pair individually.
6. **Step 3: Reducer**
   - **Input:** Pairs of lexemes with their Association vectors.
   - **Output:** Final 24-dimensional similarity vector for each pair of lexemes.
   - **Key-Value Pairs:**

        - Key: Pair of lexemes.
        - Value: 24-dimensional similarity vector.
    - **Memory Usage:** Moderate, as it calculates multiple similarity metrics for each pair.