CS 470: Data Mining

Homework II - Pattern Discovery

Prof. Kai Shu
Due at Feb. 26th 2025, 11:59 PM

Submission Instructions: Submit your assignment through the Canvas system. Upload a single ZIP archive file named $hw1\_ < yourfirstname > \_ < yourlastname > .zip$, containing the report as well as an output.txt files from Part 2 (see the details below). No email submissions are accepted. At the top of your solution, include a section named "Collaboration statement" in which you acknowledge any collaboration, help, or resource you used or consulted to complete this assignment. You can use any programming language(s) to perform the computations and visualizations.

Note: Please observe the Emory University Honor Code. An automatic plagiarism checker will be used on submissions.

For this assignment, you will conduct frequent itemset mining on the dataset provided in **data.csv**, as described below. The **data.csv** file contains a collection of tweets from 2014 related to flu shots. Each row represents an individual tweet, while each column corresponds to a specific attribute. The dataset includes 12 attributes, which are largely self-explanatory based on their names. Your primary focus should be on the *text_keywords* attribute; however, you are encouraged to utilize other attributes to support your analysis and discussion of the frequent itemset mining results in your report.

Your implementation must meet specific requirements outlined in the tasks below. The assignment is divided into three main tasks: **Code Implementation, Output File Generation, and Report Submission**, each graded separately.

1 Part 1: Algorithm Implementation (50 Points)

Your primary goal is to implement the **Apriori algorithm** for frequent itemset mining. Follow these guidelines:

- Use the `text_keywords` column from `data.csv`, where each row contains a set of keywords separated by `";"`. Each keyword is an **item**, and the set of keywords in a row is treated as a **transaction**.
- Implement the **Apriori algorithm** in Python, Java, R, or MATLAB. You may reference pseudocode from course materials or textbooks.
- You are encouraged to use optimization techniques (either your own or those introduced in class) to improve algorithm efficiency. Clearly explain these techniques in your report.
- Your program must be executable with **three command-line parameters**:

1. **Input dataset file name** (e.g., `data.csv`)
2. **Minimum support count** (an integer threshold)
3. **Output file name** (e.g., `output.txt`)

- A frequent itemset is one where the **support count** (number of transactions containing the itemset) meets or exceeds the given threshold.
- **Performance trade-off:** If the support threshold is too low, execution may take too long. If it is too high, very few itemsets may be found. Tune this parameter accordingly.
- **Execution time constraint:** Your implementation must complete in under 10 minutes when run on `data.csv` with a minimum support count of 500 on a system with at least a 2.3 GHz 8-Core Intel Core i9 CPU.

1.1 Grading for Part 1 (50 Points)

- **Full Points (50):** Code correctly implements the Apriori algorithm and meets all requirements.
- **Minor Mistakes (-10):** Small errors that do not significantly impact correctness.
- **Moderate Errors (-20):** Some mistakes present, but the program works in most cases.
- **Serious Mistakes (-30):** Major errors, but the program still functions in some cases.
- **Zero Points (0):**
  - Code does not compile according to your README instructions.
  - Program mostly produces incorrect results or crashes.
  - Execution time exceeds **10 minutes** with a **minimum support count of 500** on the specified CPU.

2 Part 2: Output File Generation (10 Points)

Your program must generate an **output file (output.txt)** containing the discovered frequent itemsets. The file should follow this format:

- Each line represents **one frequent itemset**, with items separated by a **single space**.
- The **support count** is included in parentheses at the end of the line.
- The itemsets must be **ranked in descending order** based on support count.

2.1 Example Format of `output.txt`

```
home flu shot (530)
vaccine recommended (420)
doctor flu (389)
```

A sample output file (`example-output.txt`) is provided for reference.

2.2 Grading for Part 2 (10 Points)

- **Full Points (10):** Correct format and ranking.
- **Format Errors (-3):** Incorrect format or not sorted in descending order.
- **Wrong File Name (-2):** Output file not named `output.txt`.
- **Incorrect Results (-5):** Frequent itemsets are missing or incorrect.

3 Part 3: Report Submission (40 Points)

You must submit a well-structured **PDF report** that analyzes your results and implementation. Your report should include:

1. **Chosen Minimum Support Threshold**
   - Justify the value selected for the **minimum support count** in your program.
   - Explain the trade-off between **efficiency** and the **number of frequent itemsets**.
2. **Algorithm Implementation & Optimizations**
   - Explain your **implementation strategy** and any **optimization techniques** used.
   - Provide references for any existing optimizations you applied.
3. **Results Analysis**
   - Discuss the **patterns discovered** from the dataset.
   - Explain **what insights** can be derived from the frequent itemsets found.
4. **Lessons Learned**
   - Reflect on the challenges faced during the assignment.
   - Mention any improvements you would consider for future implementations.

3.1 Grading for Part 3 (40 Points)

- **Full Points (40):** Clear, complete, and well-organized report.
- **Incomplete Analysis (-10 to -20):** Missing key sections or explanations.
- **Poorly Written (-30 to -40):** Report lacks clarity, is disorganized, or missing entirely.

4 Important Notes

- **Start Early** A poorly planned implementation may take days to run.
- **Inefficient Implementations** will receive **zero points** if they are excessively slow.
- You may study online repositories for frequent pattern mining (e.g., FIMI Repository), but **you must not copy their code**.

- **Academic Integrity:** Submissions will be checked using an automatic plagiarism detector.

Grading Criteria

- You will get full score for each task that is complete, correct, and well justified; you will receive a partial score proportional to the level of completion and correctness if the task is partially incomplete, or with minor mistakes, or not well justified; zero point if the task has major omissions, major errors, or it has wrong or totally missing justification.
- -10 points for missing collaboration statement in the situation when you should have it.