DMML Assignment 1: Report of Frequent Itemset Mining for the Bag of Words Dataset

Trishita Patra - MDS202440
Boda Surya Venkata Jyothi Sowmya - MDS202413

Introduction

This report summarizes the results of computing frequent K-itemsets for three datasets from the Bag of Words collection:

- Enron Emails: D=39,861, W=28,102, NNZ=6.4M
- NIPS full papers: D=1,500, W=12,419, NNZ=1.9M
- KOS blog entries: D=3,430, W=6,906, NNZ=467K

The Apriori algorithm was implemented in Python using Google colab to extract frequent itemsets occurring with frequency atleast F.

Methodology

1. **Data Processing**:
   - Vocabulary files (`vocab.file.txt`) were read to map word IDs to words.
   - Document-word files (`docword.file.txt`) were parsed to record word occurrences in documents.
   - Only words appearing in at least F documents were considered to build the K-item sets, since X.count >= (X,Y).count .
2. **Apriori Algorithm Implementation**:
   - Frequent 1-itemsets were identified by checking word occurrences against threshold F.
   - Candidate itemsets of increasing size (K) were generated iteratively.
   - Pruning was applied by merging itemsets that shared the same elements except for the last one.
   - Support for each candidate itemset was counted, retaining those meeting F.
3. **Execution Parameters**: The Apriori algorithm was executed with varying parameters to analyze the datasets comprehensively:
   - K : The size of frequent itemsets to find. Values tested include {2, 3, 4, ...}.
   - F : The minimum support threshold, Frequency. Values tested include {1300, 1200, 1100, ...}.
   - Size of the dataset.

Result

Table 1: Frequent Itemsets for the Enron Dataset

Dataset	K	F	Total Frequent Itemsets Found	Time Taken (s)
Enron	2	2000	18	9.55
Enron	2	1900	26	11.16

Dataset	K	F	Total Frequent Itemsets Found	Time Taken (s)
Enron	2	1800	40	12.90
Enron	2	1700	69	14.23
Enron	2	1600	117	17.29
Enron	2	1500	202	19.34
Enron	2	1400	347	20.74
Enron	2	1300	576	23.30
Enron	2	1200	964	32.79
Enron	2	1100	1531	38.40
Enron	2	1000	2578	44.42
Enron	2	900	4424	56.21
Enron	3	1500	3	22.21
Enron	3	1200	61	80.76
Enron	4	1200	2	40.47
Enron	5	800	162	221.84
Enron	5	1200	0	70.91
Enron	7	700	15	514.36
Enron	10	900	0	119.89

Table 2: Frequent Itemsets for the NIPS Dataset

Dataset	K	F	Total Frequent Itemsets Found	Time Taken (s)
NIPS	2	1300	10	0.01
NIPS	2	1200	38	0.02
NIPS	2	1100	82	0.04
NIPS	2	1000	179	0.15
NIPS	2	900	365	0.29
NIPS	2	800	724	0.80
NIPS	3	1200	36	0.05
NIPS	4	1000	362	0.78
NIPS	4	800	4660	10.51
NIPS	5	1100	18	0.22
NIPS	5	800	4521	19.18
NIPS	7	900	46	4.40
NIPS	8	800	64	30.00
NIPS	10	800	0	29.61

Table 3: Frequent Itemsets for the KOS Dataset

Dataset	K	F	Total Frequent Itemsets Found	Time Taken (s)
KOS	2	1200	1	0.01
KOS	2	1100	2	0.01
KOS	2	1000	3	0.01
KOS	2	900	6	0.03
KOS	2	800	11	0.03
KOS	2	700	23	0.05
KOS	2	600	54	0.06
KOS	2	500	101	0.11
KOS	2	350	481	0.50
KOS	3	700	1	0.06
KOS	4	550	1	0.15
KOS	4	500	5	0.24
KOS	4	350	796	1.92
KOS	5	400	5	0.86
KOS	7	350	1	3.29
KOS	10	350	0	4.27

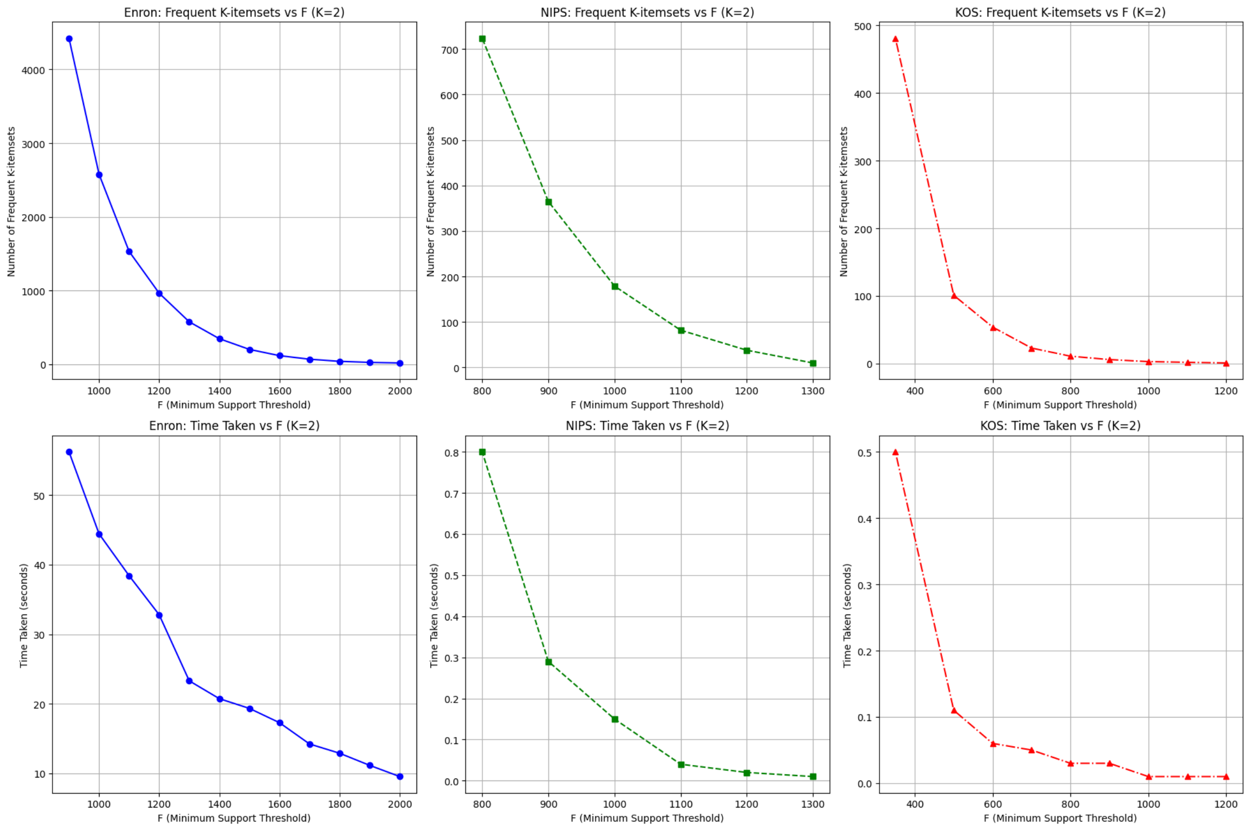
Observation

Plot 1-6 : Representing how the number of K-frequent itemsets, and time taken (in seconds) vary with frequency `F` for fixed 'K' = 2

```
import matplotlib.pyplot as plt

# Data for Enron dataset (K=2)
enron_F = [2000, 1900, 1800, 1700, 1600, 1500, 1400, 1300, 1200, 1100,
1000, 900]
enron_time = [9.55, 11.16, 12.90, 14.23, 17.29, 19.34, 20.74, 23.30,
32.79, 38.40, 44.42, 56.21]
enron_freq = [18, 26, 40, 69, 117, 202, 347, 576, 964, 1531, 2578,
4424]

# Data for NIPS dataset (K=2)
nips_F = [1300, 1200, 1100, 1000, 900, 800]
nips_time = [0.01, 0.02, 0.04, 0.15, 0.29, 0.80]
nips_freq = [10, 38, 82, 179, 365, 724]

# Data for KOS dataset (K=2)
kos_F = [1200, 1100, 1000, 900, 800, 700, 600, 500, 350]
kos_time = [0.01, 0.01, 0.01, 0.03, 0.03, 0.05, 0.06, 0.11, 0.5]
kos_freq = [1, 2, 3, 6, 11, 23, 54, 101, 481]
```

```
# Data for Enron (F = 1200)
enron_k = [2, 3, 4, 5]
enron_time_taken = [32.79, 80.76, 40.47, 70.91]  # Time Taken
(seconds)
enron_frequent_itemsets = [964, 61, 2, 0]  # Total Frequent Itemsets
Found

# Data for NIPS (F = 800)
nips_k = [2, 4, 5, 8, 10]
nips_time_taken = [0.80, 10.51, 19.18, 30.00, 29.61]  # Time Taken
(seconds)
nips_frequent_itemsets = [724, 4660, 4521, 64, 0]  # Total Frequent
Itemsets Found

# Data for KOS (F = 350)
kos_k = [2, 4, 7, 10]
kos_time_taken = [0.50, 1.92, 3.29, 4.27]  # Time Taken (seconds)
kos_frequent_itemsets = [481, 796, 1, 0]  # Total Frequent Itemsets
Found

# Create a 2x3 grid of subplots for time taken
fig, axes = plt.subplots(2, 3, figsize=(18, 12))  # 2 rows, 3 columns

# Plot 1: Enron - Frequent K-itemsets vs F
axes[0, 0].plot(enron_F, enron_freq, marker='o', linestyle='-',
color='blue')
axes[0, 0].set_title('Enron: Frequent K-itemsets vs F (K=2)')
axes[0, 0].set_xlabel('F (Minimum Support Threshold)')
axes[0, 0].set_ylabel('Number of Frequent K-itemsets')
axes[0, 0].grid(True)

# Plot 2: NIPS - Frequent K-itemsets vs F
axes[0, 1].plot(nips_F, nips_freq, marker='s', linestyle='--',
color='green')
axes[0, 1].set_title('NIPS: Frequent K-itemsets vs F (K=2)')
axes[0, 1].set_xlabel('F (Minimum Support Threshold)')
axes[0, 1].set_ylabel('Number of Frequent K-itemsets')
axes[0, 1].grid(True)

# Plot 3: KOS - Frequent K-itemsets vs F
axes[0, 2].plot(kos_F, kos_freq, marker='^', linestyle='-.',
color='red')
axes[0, 2].set_title('KOS: Frequent K-itemsets vs F (K=2)')
axes[0, 2].set_xlabel('F (Minimum Support Threshold)')
axes[0, 2].set_ylabel('Number of Frequent K-itemsets')
axes[0, 2].grid(True)

# Plot 1: Enron - Time vs F
axes[1, 0].plot(enron_F, enron_time, marker='o', linestyle='-',
```

```
color='blue')
axes[1, 0].set_title('Enron: Time Taken vs F (K=2)')
axes[1, 0].set_xlabel('F (Minimum Support Threshold)')
axes[1, 0].set_ylabel('Time Taken (seconds)')
axes[1, 0].grid(True)

# Plot 2: NIPS - Time vs F
axes[1, 1].plot(nips_F, nips_time, marker='s', linestyle='--',
color='green')
axes[1, 1].set_title('NIPS: Time Taken vs F (K=2)')
axes[1, 1].set_xlabel('F (Minimum Support Threshold)')
axes[1, 1].set_ylabel('Time Taken (seconds)')
axes[1, 1].grid(True)

# Plot 3: KOS - Time vs F
axes[1, 2].plot(kos_F, kos_time, marker='^', linestyle='-.',
color='red')
axes[1, 2].set_title('KOS: Time Taken vs F (K=2)')
axes[1, 2].set_xlabel('F (Minimum Support Threshold)')
axes[1, 2].set_ylabel('Time Taken (seconds)')
axes[1, 2].grid(True)

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



From the above plots we extract that,

- With fixed K, as F decreases, the number of frequent itemsets grows exponentially, and the time taken increases accordingly.
- For a given `(K,F)` pair, the larger the dataset, the higher is the run time.

The runtime increases significantly as K (itemset size) increases and F (minimum support threshold) decrease because:

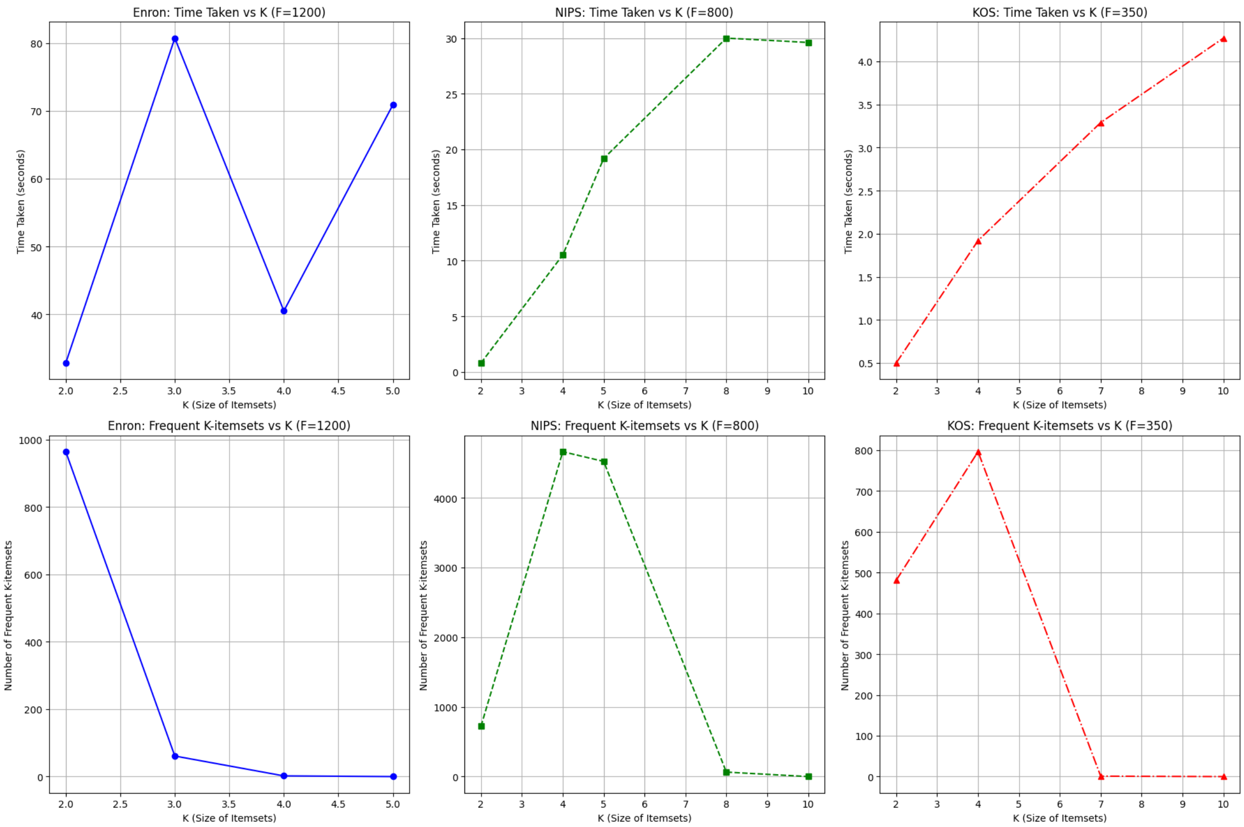
- **Larger** K => More combinations of itemsets need to be generated and checked.
- **Smaller** F => More itemsets meet the minimum support threshold, increasing the number of candidates to process.

Plot 7-12 : Representing how the Number of K-frequent itemsets, and time taken (in seconds) vary with size of itemset `K` for fixed frequency `F`.

- `F` = 1200 for Enron
- `F` = 800 for Nips
- `F` = 350 for Kos

```
# Create a 2x3 grid of subplots for number of frequent K-itemsets
fig, axes = plt.subplots(2, 3, figsize=(18, 12))  # 2 rows, 3 columns

# Plot 4: Enron - Time vs K
axes[0, 0].plot(enron_k, enron_time_taken, marker='o', linestyle='-',
color='blue')
```

```python
axes[0, 0].set_title('Enron: Time Taken vs K (F=1200)')
axes[0, 0].set_xlabel('K (Size of Itemsets)')
axes[0, 0].set_ylabel('Time Taken (seconds)')
axes[0, 0].grid(True)

# Plot 5: NIPS - Time vs K
axes[0, 1].plot(nips_k, nips_time_taken, marker='s', linestyle='--',
color='green')
axes[0, 1].set_title('NIPS: Time Taken vs K (F=800)')
axes[0, 1].set_xlabel('K (Size of Itemsets)')
axes[0, 1].set_ylabel('Time Taken (seconds)')
axes[0, 1].grid(True)

# Plot 6: KOS - Time vs K
axes[0, 2].plot(kos_k, kos_time_taken, marker='^', linestyle='-.',
color='red')
axes[0, 2].set_title('KOS: Time Taken vs K (F=350)')
axes[0, 2].set_xlabel('K (Size of Itemsets)')
axes[0, 2].set_ylabel('Time Taken (seconds)')
axes[0, 2].grid(True)

# Plot 4: Enron - Frequent K-itemsets vs K
axes[1, 0].plot(enron_k, enron_frequent_itemsets, marker='o',
linestyle='-', color='blue')
axes[1, 0].set_title('Enron: Frequent K-itemsets vs K (F=1200)')
axes[1, 0].set_xlabel('K (Size of Itemsets)')
axes[1, 0].set_ylabel('Number of Frequent K-itemsets')
axes[1, 0].grid(True)

# Plot 5: NIPS - Frequent K-itemsets vs K
axes[1, 1].plot(nips_k, nips_frequent_itemsets, marker='s',
linestyle='--', color='green')
axes[1, 1].set_title('NIPS: Frequent K-itemsets vs K (F=800)')
axes[1, 1].set_xlabel('K (Size of Itemsets)')
axes[1, 1].set_ylabel('Number of Frequent K-itemsets')
axes[1, 1].grid(True)

# Plot 6: KOS - Frequent K-itemsets vs K
axes[1, 2].plot(kos_k, kos_frequent_itemsets, marker='^',
linestyle='-.', color='red')
axes[1, 2].set_title('KOS: Frequent K-itemsets vs K (F=350)')
axes[1, 2].set_xlabel('K (Size of Itemsets)')
axes[1, 2].set_ylabel('Number of Frequent K-itemsets')
axes[1, 2].grid(True)

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



From the above plots we extract that,

- With F fixed, and increasing K, combinations of K-itemset grows rapitdly, but the number of frequent itemsets sstarts decreasing after one point, since chances of all the words of a K-itemset being in the same document decreases.
- Usually, runtime increases with increasing K, since the algorithm has to consider higher number of K-itemsets, but the drop in runtime, for high K, e.g. K =4, F =1200 for enron dataset is due to pruning in apriori algorithm.

Conclusion

Time Complexity $O(2^W)$:

- Increases exponentially with K due to combinatorial growth of candidate itemsets.
- Increases as F decreases because more itemsets meet the support threshold.
- Larger datasets (e.g., ENRON) take significantly longer to process than smaller ones (e.g., NIPS, KOS).

The Apriori algorithm has two main phases:

- Counting support for candidate itemsets .
- Generating candidate itemsets **iteratively**:
  - In the worst case scenario, contributes time complexity of
$$O(2^W)$$
    where W is the number of words in the dataset.

Space Complexity:

- Depends on the number of frequent itemsets and intermediate candidate itemsets.
- Larger datasets and lower F values require more memory to store frequent itemsets and intermediate results.

The space complexity depends on:

- Storing the dataset `word_docs`.
  - `word_docs` stores a set of document IDs for each word:
  - If there are $ W $ words and $ D $ documents, the space required is:

$$O(W \cdot D)$$

- Storing frequent itemsets and candidate itemsets:
  - `freq_itemsets` stores frequent itemsets and their corresponding document sets.
  - If there are $ N_k $ frequent itemsets at level $ k $, the space required is proportional to $ N_k $.
  - `candidates` temporarily stores candidate itemsets.
  - In the worst case, the number of candidates grows combinatorially, requiring:

$$O(2^W)$$

- Temporary storage during intersection operations:
  - For each candidate itemset of size $ k $, the intersection requires temporary storage for $ k $ document sets.
  - If the average size of a document set is $ S $, the space required is:

$$O(k \cdot S)$$

Hence, Combining all components, and considering $ (k \cdot S) <= (W \cdot D) $ the overall space complexity is:

$$O(W \cdot D + 2^W)$$