Application of a Mining Algorithm to Finding Frequent Patterns in a Text Corpus: A Case Study of the Arabic

Imran Ali*

*Computer Science & Information Management Program, Asian Institute of Technology, PO Box 4, Klong Luang, Pathumthani 12120, Thailand
Ali.Imran@ait.ac.th

Abstract

Information repositories containing text data of different languages are abundant on the World Wide Web. Digital corpora of sacred text of Islam related to Quran containing Arabic language are also publicly available. The availability of these corpora and intelligent application to analyze them are vital to better comprehend the religious text of Islam. In this paper I propose a method of representing the Quranic text corpus as a graph, and apply a frequent sub-path mining algorithm on it to generate frequent patterns. I have explained how the resulting frequent patterns can be used for subjective indexing and clustering similar verses of Quran.

Keywords: FS, Apriori, Quran, data mining, frequent subpath, verse similarity

1. Introduction

The Internet which started as a collection of static websites has now become a digital globe containing text, audio, video, and image repositories of huge volume. The correlated advancement of computing and communication technology has exploded the digital globe with data. The need to transform this data into useful information gave birth to field called data mining. Data mining has introduced tools and techniques to extract interesting patterns from huge data sets. The most widely known application of data mining is market basket analysis. This application analyzes the data set of supermarket by finding association between the items sold together. Many systems now recommend products or services based on frequent patterns e.g. they recommend books, touring a place, or multimedia contents.

Frequent pattern, as the name suggests are patterns that frequently appear in a data collection. Itemsets, subsequences, or substructures are different terms used to refer to patterns of different kind. The most common pattern called frequent Itemset refers to set of items that appear together frequently in a data set. Example of itemsets that may appear in the transactional data set of stationery may be pencil, paper, marker etc. A substructure can refer to various structural forms e.g. subgraphs, subtrees etc. Mining frequent patterns can reveal useful association rules and correlations within data e.g. to determine items that are frequently purchased together.

1.1. Problem Statement

The problem of mining frequent patterns from text data can be split into two parts, sophisticated data structures which represent data without losing important information such as bag of words representation, and efficient algorithm that operates on the given data structure to mine frequent patterns.
1.2. Proposed Solution

The basic idea is to represent Quranic text using a complete graph, and apply an efficient frequent pattern mining algorithm which can take advantage of the graph representation.

1.2. Significance of Mining Frequent Patterns

Research work of Quranic scholars for the past fifteen centuries has been limited to their personal knowledge and familiarity of the Quran. Availability of computation resources and tools provides the opportunity to explore new possibilities of discovering new information hidden in religious text repositories. Frequent patterns obtained from Quran can serve as the knowledge base for many useful applications including:

1. Automated subjective index: The discussion of a specific topic in Quran is not completely covered in a single chapter. Different aspects of a single topic can reappear in many chapters. Therefore frequent patterns can be used to construct a subjective index where all verses on a single topic can be collected.

2. Clustering: All verses containing a given frequent pattern form a cluster. Such clusters may reveal interesting characteristics important in determining the location, event and the year of revelation.

3. Verse Similarity: Numerous verses of Quran are similar to each other in various ways; some verses are exactly identical to each other, emphasizing the importance of the text. A segment of a verse may be phonetically similar to a segment of another verse; it is difficult for a person reciting verbatim to distinguish between such segments. An application that provides a complete index of phonetically similar verses with their respective location, and the utility to mark frequent mistakes a reciter commits can assist in the process of memorizing.

4. Quran-Hadith linkage: Numerous collections of Hadith (the sayings of prophet) explain verses of Quran. An application to discover the linkage between Quran and Hadith can help in better understanding of the verses of Quran. Retrieval of Hadith relevant to a keyword can be improved by expanding a keyword to all frequent patterns containing it.

In this paper I propose a method of finding frequent patterns using AFS and use this method to extract frequent patterns from Quran. Before presenting the potential applications that can benefit by mining this data set, I describe few important characteristic of Quran. Muslims believe that Quran is a divine book revealed to Muhammad through the angel Gabriel over a period of twenty-three years. Quran contains one hundred fourteen chapters of different length. Each chapter contains verses called ayah. Scholar and researchers have spent years on various classifications of the verses and chapters of Quran using different parameters. One of the major reasons of such classification is to understand different dimensions of a single topic.

This paper is organized as follows. Section 2 explains the working of AFS algorithm. Section 3 contains experimental results conducted to compare the performance of AFS algorithm vs. apriori algorithm. Section 4 details the application of AFS to Quranic corpus. Section 5 discusses the results obtained by employing AFS to first four chapters of the Quranic corpus. Conclusion and future work is presented in section 6.
1.4. Related Work

A classic algorithm for mining association rules from sales transactions called apriori was proposed in \(^1\). This algorithm attempts to find subsets of the itemsets, which are common to at least a minimum number, called the confidence threshold. Apriori uses breadth-first search and a hash tree structure to count candidate item sets efficiently. Apriori repeatedly scans the database to count candidates, and assumes it to be memory residents. If the size of the transactions in database is long, which is usually true for practical applications, the algorithm is vulnerable to exponential time worst-case behaviour. Since the performance of the apriori algorithm is dictated by its support counting procedure, most research has focused on that aspect of the algorithm. Related research on improving the efficiency of the apriori algorithm include \(^2\), \(^3\), and \(^4\) amongst others. An in depth survey on frequent pattern mining by \(^5\) compare different itemset mining algorithms. An interesting conclusion drawn by the author in the survey states that different implementations of the same algorithm vary in performance.

Research work on apriori has not been limited to performance enhancement for itemset mining; many apriori-based algorithms have also been proposed for mining frequent patterns from sophisticated structures like graphs. \(^6\) details a survey of these algorithms.

AFS (Apriori for Frequent Subpaths) is a frequent subpath mining algorithm proposed in \(^7\). It uses a vertex based candidate generation method that increases the subpath size by one vertex at each iteration of the algorithm. AFS search for frequent subpaths starting with frequent vertices or frequent subpaths of length zero, and proceeds in a bottom-up manner by generating candidates having an extra vertex. The main reason for preferring AFS over apriori is its ability to exploit the underlying graph structure and reducing the run time complexity from exponential to quadratic. Research literature on the application of computing to Quranic data sets includes a variety of attempts. Following paragraphs cite some of the related articles.

An open source tool called aConcorde was proposed in \(^8\) which displays a target word in its context (verse containing the word). aConcorde calculates similarity between verses by using cosine function. The context for the target word can be set to display certain number of words before and after the target word. The resulting verses containing the target word are displayed without any information about their location. Furthermore the search is limited to single word, which means that sequences containing two or more words cannot be searched. In \(^9\) a method of discovering thematic interrelationship between different chapters of Quran using their lexical frequency profiles was proposed. The author experimented with twenty four chapters of length more than one thousand words. Results obtained by experiments supported the proposed methodology. Development of a comprehensive Quranic Knowledge Map to provide on-line applications, including morphological searching, interlinear translation, word sense disambiguation, and concept topic map among others was proposed in \(^10\). While linguistic features proposed are available at corpus.quran.com other proposed features such as Quran to hadith linkage, automated question answering were not available till the writing of this paper.

2. AFS Algorithm

As mentioned in the section 1.4, AFS algorithm shares similar characteristics with the apriori algorithm for itemset mining, however performs much better than apriori by exploiting graph structure. I present here a sample run of the algorithm to demonstrate how it mines frequent pattern. Figure 1 shows a 5 by 4 grid graph with twenty vertices labeled a through t. This graph is completely represented by a set of vertices, and its adjacency information. Each vertex keeps a record of its adjacent vertices e.g. the vertex labeled g has four vertices in its
adjacency list b; f; h; and l. I assume a minimum support count of 2 and that the transaction database contains three paths shown in Figure 1.

2.1. Frequent 0-Subpaths

The algorithm starts by scanning the transaction database of paths, calculating the support count for each vertex in the graph. Every vertex is used to generate candidate frequent subpath of length zero i.e. subpaths having only one vertex. After generating candidate zero subpaths the support for each candidate is checked against a given minimum threshold. All candidates below support threshold(paths containing vertices a; b; c; d; e; j; k; l; o; p; q; r; s; t) are discarded . The remaining candidates form the frequent 0-subpaths.

2.2. Candidate Generation

Frequent 0-subpaths are extended to generate candidate 1-subpath; last vertex of each frequent-0 subpath is paired with each vertex in its adjacency. Thus frequent 0-subpath having vertex f, generates three candidate 1-subpaths, (f; a); (f; g); (f; k). The algorithm prunes the candidates (f; a) and (f; k) because frequent 0-subpaths having vertex a, and k do not exist.

2.3. Support Checking

The support checking routine initializes the count of each candidate to zero. The transaction database containing paths is scanned and the count is incremented as each occurrence of the candidate is encountered.

![Figure 1: A Sample Grid Graph with Three Paths](image)

3. Experimental Results

I have conducted the following experiments to verify the theoretic claim in7 that the runtime complexity of AFS is quadratic whereas apriori is vulnerable to exponential worst case behavior. I have obtained test data by generating grid graphs and complete graphs of different size with random paths. Both algorithms were implemented in Java programming language. The experiments were conducted on DELL vostro laptop having Intel Core 2 Duo 2.00 GHz processor with 2 GB RAM. A constant minimum support count of 2 was used for both algorithms.
3.1. Grid Graph

Grid graph of four different dimensions (row by column) were generated. To ensure that the randomly generated paths coincide, a random seed of a given size was first generated. For each dimension the experiments were repeated three times. Table 1 shows the comparison of the two algorithms in terms of average execution time, and the average of candidates generated by both algorithm.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Probability</th>
<th>Average Execution Time</th>
<th>Average Candidates Generated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Apriori</td>
<td>AFS</td>
</tr>
<tr>
<td>10</td>
<td>0.25</td>
<td>74</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>0.50</td>
<td>86</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>0.75</td>
<td>286</td>
<td>21</td>
</tr>
<tr>
<td>15</td>
<td>0.25</td>
<td>125</td>
<td>26</td>
</tr>
<tr>
<td>15</td>
<td>0.50</td>
<td>567</td>
<td>23</td>
</tr>
<tr>
<td>15</td>
<td>0.75</td>
<td>1628</td>
<td>17</td>
</tr>
<tr>
<td>20</td>
<td>0.25</td>
<td>420</td>
<td>21</td>
</tr>
<tr>
<td>20</td>
<td>0.50</td>
<td>1474</td>
<td>22</td>
</tr>
<tr>
<td>20</td>
<td>0.75</td>
<td>40873</td>
<td>32</td>
</tr>
<tr>
<td>25</td>
<td>0.25</td>
<td>424</td>
<td>212</td>
</tr>
<tr>
<td>25</td>
<td>0.50</td>
<td>3149</td>
<td>217</td>
</tr>
<tr>
<td>25</td>
<td>0.75</td>
<td>2375727</td>
<td>213</td>
</tr>
</tbody>
</table>

3.2. Complete Graph

Complete graphs were obtained by a simple random process. For each pair of vertices in the graph a random number was generated, if the random number drawn was less than the probability assumed an edge was added to the graph. Probability values of 0.25, 0.50, and 0.75 were used. Table 2 shows the comparison of the two algorithms in terms of average execution time and average number of candidates generated.

<table>
<thead>
<tr>
<th>Dimension of Grid</th>
<th>Average execution time in milliseconds</th>
<th>Average candidates generated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Apriori</td>
<td>AFS</td>
</tr>
<tr>
<td>5 x 5</td>
<td>1603</td>
<td>43</td>
</tr>
<tr>
<td>5 x 10</td>
<td>3221</td>
<td>62</td>
</tr>
<tr>
<td>5 x 15</td>
<td>2473</td>
<td>87</td>
</tr>
<tr>
<td>5 x 20</td>
<td>62917</td>
<td>140</td>
</tr>
</tbody>
</table>
The results obtained highlight that the AFS algorithm outperforms Apriori algorithm by efficiently generating the frequent candidates, taking advantage of the graph structure.

4. AFS Application to Quranic Arabic Text

The block diagram in Figure 2 depicts the steps involved in modeling Quranic corpus as a complete graph. The complete graph G that represents the Quranic corpus consists of a set of vertices V and a set of edges E between vertices. Each element \( v_0 \in V \) is a unique numeric identifier corresponding to a token. Tokens have been generated using java API provided by

![Block Diagram of the Proposed Methodology](image)

**Figure 2. Block Diagram of the Proposed Methodology**

Each element \( e \in E \) is a tuple \((v1, v2)\) indicating that \( v2 \) is adjacent to \( v1 \), where \( v_1 \) and \( v_2 \) \( \in V \). The set of Edges E has been acquired by computing all distinct bigrams in the corpus. The transaction database D consists of tuples \((L, Q)\), where \( L \) denotes the location of a verse by an ordered pair \((ch, vrs)\); \( ch \) and \( vrs \) denote chapter and verse number respectively, and \( Q \) is sequence \((v_0, v_1, \ldots, v_k)\) of vertices representing the verse of Quran. Figure 3 shows a sample graph representation of the first three verses of the first chapter. In Figure 3 vertices are drawn as circles, the numeric identifiers in each circle correspond to the tokens, and the lines connected circles represent sequences of adjacent tokens in a verse. The transaction database D is shown in Table 3.

<table>
<thead>
<tr>
<th>L(ch, vrs)</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 1</td>
<td>14921 18993 18534 17259</td>
</tr>
<tr>
<td>1, 2</td>
<td>18621 18912 18891 18850</td>
</tr>
<tr>
<td>1, 3</td>
<td>18534 17259</td>
</tr>
</tbody>
</table>
Given the sample graph of Figure 3 and the transaction database shown in Table 3, AFS algorithm will mine all frequent subpaths. For a minimum support count of 2 the AFS algorithm will generate one frequent subpath (18534 17259). The numeric identifiers in the resulting frequent subpath can be replaced by respective tokens which would result in third verse of chapter one. The AFScheckSupport routine besides checking the minimum support from the database also records the location (chapter and verse number) so that all verses containing a frequent pattern can be retrieved.

![Figure 3 Graph Representation of the Data Shown in Table 3](image)

5. Results

I have applied the proposed method to first four chapters to generate frequent subpaths. There are various reasons for selecting the initial chapters. The text of first four chapters of Quran form one of the seven approximately equal parts (called manazil). The verses of the first four chapters are lengthier as compared to the last chapters, and therefore, it is difficult for a reader to observe a frequent pattern manually. I have developed a user interface that allows to easily sort frequent patterns on different features (length of the pattern, count or frequency of the pattern, alphabetic order).

5.1. Subject Index

Part of speech tags of frequent patterns consisting of one word can be used to create subject index. All such patterns corresponding to the tags noun, proper noun and adjective can be filtered to form subject index.

5.2. Verse Similarity

A verse may contain one or more frequent patterns of different length. The longest frequent pattern can be used to measure lexical similarity of all verses it refers to. Most of the methods proposed in literature for finding similar verses of Quran reduce words to their root. As a result of word to root normalization, there is no clean separation between phonetically similar and dissimilar verses.
6. Conclusion and Future Work

In this paper I have presented how AFS can be used to mine patterns from Quranic text and showed that frequent patterns generated by the algorithm cluster similar or identical verses. I plan to extend our work by utilizing these frequent patterns for knowledge discovery in larger text corpus such as hadith and commentaries of Quran.

References