SS-ZG548: ADVANCED DATA MINING

15 Mining with Graph Data

Dr. Kamlesh Tiwari
Assistant Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

Oct 28, 2021 ONLINE (WILP @ BITS-Pilani July-Dec 2021)

http://ktiwari.in/adm

Graph Mining

- Where is graph data? 1) Chemical Data, 2) Computational Biology, 3) Social Networking, 4) Web Links, 5) Computer networks

[1] Beeferman, Doug and Berger, Adam, "Agglomerative clustering of a search engine query log", in ACM SIGKDD international conference on Knowledge discovery and data mining, pages=407–416, ACM 2000

Graph Mining

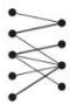
- Where is graph data? 1) Chemical Data, 2) Computational Biology, 3) Social Networking, 4) Web Links, 5) Computer networks
- What we want to do? 1) Query and Indexing for Graphs, 2) Graph Matching, 3) Keyword Search in Graphs, 4) Graph Clustering and Subgraph Extraction, 5) Graph Classification, 6) Frequent Pattern Mining in Graphs, 7) Streaming algorithms of Graphs, 8) Privacy Preserving data mining of graphs, 9) Web applications *etc*

[1] Beeferman, Doug and Berger, Adam, "Agglomerative clustering of a search engine query log", in ACM SIGKDD international conference on Knowledge discovery and data mining, pages=407–416, ACM 2000

Graph Mining

- Where is graph data? 1) Chemical Data, 2) Computational Biology, 3) Social Networking, 4) Web Links, 5) Computer networks
- What we want to do? 1) Query and Indexing for Graphs, 2) Graph Matching, 3) Keyword Search in Graphs, 4) Graph Clustering and Subgraph Extraction, 5) Graph Classification, 6) Frequent Pattern Mining in Graphs, 7) Streaming algorithms of Graphs, 8) Privacy Preserving data mining of graphs, 9) Web applications *etc*
- Difficult to have complete graph in memory[1]

- **Graph Clustering:** Divide the graph clusters based on similarity. Not all graphs have a structure.
- **Bipartite Graph Clustering:** Search-url maping



[1] Beeferman, Doug and Berger, Adam, "Agglomerative clustering of a search engine query log", in ACM SIGKDD international conference on Knowledge discovery and data mining, pages=407–416, ACM 2000

Bipartite Graph Clustering with BB Algorithm

- Similarity measure

$$\sigma(x,y) = \begin{cases} \frac{|N(x)\cap N(y)|}{|N(x)\cup N(y)|} & \text{if } |N(x)\cup N(y)| > 0 \\ 0 & \text{otherwise} \end{cases}$$

  where $N(x)$ is number of nodes linked to x
- Merge two nodes for which similarity is maximum from set A and repeat it for Set B
- Alternately merge till similarity is greater than a threshold

Large Dense Subgraphs in Massive Graph

- Finding large dense subgraphs in massive graphs
- Application - link spam[2]: pages may be refereed without any aim.
- Informally, a dense bipartite subgraph is a pair of subset $A, B \subseteq V$ of nodes such that $(a,b) \in E$ for 'most' $a \in A$, $b \in B$; here, 'most' parametrizes the density of the subgraph
- **Shingling** is wide usage to estimate the similarity of web pages using a particular feature extraction scheme based on overlapping windows of terms ("shingles")
- Given a subset S of a universe U of elements, generate a constant-size fingerprint such that two subsets A and B may be compared by simply comparing their fingerprints
- Similarity is Jaccard Coefficient

$$\frac{|A\cap B|}{|A\cup B|}$$

[2] Gibson, David and Kumar, Ravi and Tomkins, Andrew, "Discovering large dense subgraphs in massive graphs", In International conference on Very large data bases, pages=721–732, VLDB 2005

Large Dense Subgraphs in Massive Graph

- if π is a random permutation of the elements in the ordered universe U from which A and B are drawn, then

$$Pr[\pi^{-1}(\min_{a\in A}\pi(a)) = \pi^{-1}(\min_{b\in B}\pi(b))] = \frac{|A\cap B|}{|A\cup B|}$$

- Compute the fingerprint of A by fixing a constant number c of permutations $\pi_1, ..., \pi_c$ of U, and producing a vector whose i-th element is $\min_{a\in A}\pi_i(a)$. Similarity between two sets is estimated by number of positions their respective fingerprint agree
- We may consider every s-element set contained entirely within either set, and measure similarity by the fraction of these s-element subsets that appear in both
- This is identical to measuring the similarity of A and B by computing the Jaccard coefficient of two sets As and Bs, where $As = a_1, a_2, ..., a_s | a_i \in A$, and Bs are defined likewise

The Shingling Algorithm

Algorithm Shingle$(a_1, \ldots, a_n, s, c)$

Let H be a hash function from strings to integers
Let p be a large random prime (say, 32 bits)
Let $a_1, b_1, \ldots, a_c, b_c$ be random integers in $[1 \ldots p]$
For $i = 1$ to n do $x_i = H(\text{“}a_i\text{”})$
For $j = 1$ to c do
  For $i = 1$ to n do $y_i = (a_j * x_i + b_j) \bmod p$
  Let $y'_1, \ldots, y'_s$ be s minimal elements of y
  Let $z_j = H(\text{“}y'_1 \circ \cdots \circ y'_s\text{”})$
Output $z_1, \ldots, z_c$

- Each set will represent the outlinks of a particular node, and two nodes will be considered similar if they share many outlinks
- Employ exactly the same formulation recursively to determine the similarity between two shingles in terms of the set of nodes that contain each shingle

Discovering Large Dense Graphs

Phase I

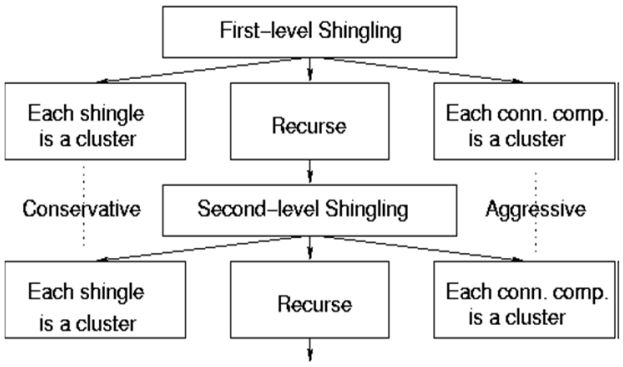
- Algorithm seeks clusters of nodes that tend to link to the same destinations for each node in the graph, it applies an (s, c) shingling algorithm to the set of destinations linked to from that node resulting in c shingles per node
- For each distinct shingle created by this process, it produces a list of all nodes containing that shingle
- This brings together the nodes that share a particular shingle (and therefore a particular set of s outlinks)
- In order to begin clustering nodes together, find sets of nodes that jointly share a sufficiently large number of shingles

Discovering Large Dense Graphs

Phase II

- perform one further level of analysis, grouping together shingles that tend to occur on the same nodes,
- so that we may use these sets of commonly co-occurring shingles as the defining patterns of a dense subgraph
- Such an analysis is yet another application of the (s, c) shingling algorithm, this time to the set of nodes associated with a particular shingle, and results in bringing together shingles that have significant overlap in the nodes on which they occur.
- If necessary, the sequence of operations may be repeated for as many levels as required
- In practice, two levels of this algorithm suffice to convert dense subgraphs of arbitrary size (i.e., hundreds or hundreds of millions of nodes) into small-size fingerprints that can then be recognized in a straightforward manner. The particular density threshold captured by the scheme is determined by the parameters s and c

Discovering Large Dense Graphs



Thank You!

Thank you very much for your attention!

Queries ?