Graph Clustering in Crime Data

Project Report

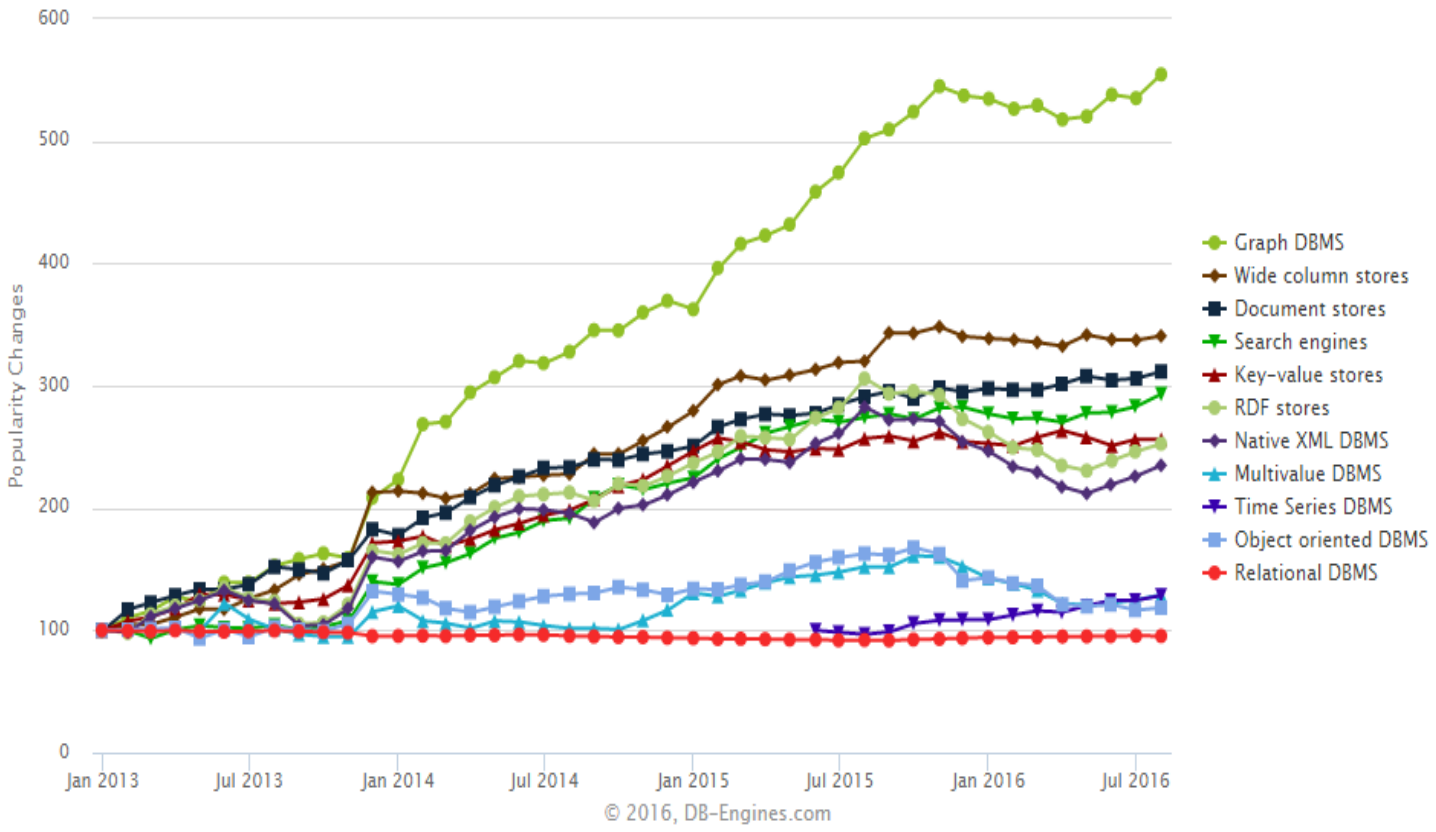
By

Saptarashmi Bandyopadhyay
510514006
Abhishek Das
510514046
Imbisat Makhdoomi
510514063

The Law of Relational Database

If the only tool you have is a relational database, everything looks like a table.

Popularity of graph database



Stanford Network Analysis Project (SNAP)

- general purpose network analysis and graph mining library
- Snap.py is a Python interface for SNAP in C++.
- A collection of more than 50 large network datasets from tens of thousands of nodes and edges to tens of millions of nodes and edges.
- Citation Networks – only research articles citation

Crime Dataset Features

- Web Source : *www.iamnirbhaya.me*
- A Web Platform to crowd source media reported crimes of Violence Against Women and Children in India
  - Survivors can visually understand the extent and realise that they are not alone. it will motivate them to report what happened to them and seek justice.
  - Common public, Government, NGOs, Funding Agencies, Media can visualise the extent of the reports and also download reports of all cases for research, campaign, advocacy and problem solving.
- Today the platform hosts information about **10,000 Documented media reports** of Violence Against Women and children in India that have been crowdsourced by volunteers.

Crime Dataset Features

- News Collection from online News sites
- Crawled and then text formatted
- Crime Categories – Dowry Harassment, Murder etc.
- State wise Collection of crime news for different crime categories on Women and Child
- Number of Text files – 32 <our dataset>

Task

- Feature extraction from Crime News
- Organize each crime event as a nodes in a graph
- Application of Graph Clustering algorithm for community detection

Only Search is not “Research”!

Sentence Level Feature Identification – The Preliminary Study

- 25 Sample sentences identified from the corpus
- Entities and Relations between entities were manually identified

- Bohuna Das had raped and murdered the woman at Garu in West Siang district on August 29, this year and an FIR in connection with it was lodged at Likabali police station the next day.
- Entity 1:Bohuna Das
- Entity 2: Women
- Entity 3: Likabal police station
- Relation(1,2): raped and murdered
- Relation(2,3): An FIR was lodged in Police station

Sentence Level Feature Identification – The Preliminary Study

- Observations:
- No general pattern on entities and relations for the different sentences
- In many sentences, more than one entity and more than one relation could be manually identified
- Identification of entity and relation features necessary

Sentence Level Feature Identification – The Preliminary Study

- Another set of 25 sentences randomly identified from the corpus
- Entities, Relations between entities and attributes of entities and relations were manually identified

An Example

- 7. "The trio had allegedly assaulted the uncle after accusing him of being a militant and molested the girl inside the auto rickshaw in which they were travelling home.“
- entity-1: The trio
- entity-2: the uncle
- attribute: travelling home
- entity-3: the girl
- attribute: travelling home
- relation[1-2]: allegedly assaulted
- relation[1-2]: accusing
- attribute: of being a militant
- relation[1-3]: molested
- attribute: inside the auto rickshaw

Sentence Level Feature Identification – The Preliminary Study

- Observations:
- Attributes of entity and relation as features
- No general pattern on entities and relations for the different sentences
- In many sentences, more than one entity and more than one relation could be manually identified
- Some general pattern observed when the verbs are similar, more so for crime related verbs
  - attacked, murdered, killed etc.

Design decisions based on the Preliminary Study

- Sentences with verbs related to crimes will be considered
- Entities refer to the perpetrator – one who commits the crime and the victim
- Relations refer the crime
- More attributes or features are available for relations and less so for the entities
- Both Entities and Relations are to be modeled as Graph Nodes in the Graph model

Design decisions based on the Preliminary Study

- Entities, Relations and Attributes of entities and relations can be modeled as the output of the Dependency parser
- Stanford Dependency Parser
- The main verb or the root of the parse tree forms the relation
- The subject of the main verb -> perpetrator
- The object of the main verb -> victim
- Dependency relations like nmod, advcl (with the verb) -> crime features
- Dependency relations like amod, nmod with the subject and object -> perpetrator or victim features

Work Plan

- Identification of information nodes for each crime event
- Node 1: crime and the relevant extracted features (Relation Node)
- Node 2: who has committed the crime and relevant extracted features (Entity Node)
- Node 3: on whom the crime has been committed and relevant extracted features (Entity Node)
- Edges : Node 2 -> Node 1 and Node 3 -> Node 1
- All the R nodes are connected forming the crime graph

Implementation Plan

- Application of Stanford Dependency Parser on each sentence
- Identification of some crime related verbs – attacked, murdered, killed etc.
- Feature extraction of Node Metadata
- Graph Modeling in – Edge List Format
- Graph Clustering in SNAP – Clauset-Newman-Moore Hierarchical Agglomeration Algorithm

Example 1

- “An enraged Danniah attacked Ashok with an axe, inflicting grievous injuries on him.

- ((u'attacked', u'VBD'), u'nsubj', (u'Danniah', u'NNP'))
- ((u'Danniah', u'NNP'), u'det', (u'An', u'DT'))
- ((u'Danniah', u'NNP'), u'compound', (u'enraged', u'NNP'))
- ((u'attacked', u'VBD'), u'dobj', (u'Ashok', u'NNP'))
- ((u'Ashok', u'NNP'), u'nmod', (u'axe', u'NN'))
- ((u'axe', u'NN'), u'case', (u'with', u'IN'))
- ((u'axe', u'NN'), u'det', (u'an', u'DT'))
- ((u'attacked', u'VBD'), u'advcl', (u'inflicting', u'VBG'))
- ((u'inflicting', u'VBG'), u'dobj', (u'injuries', u'NNS'))
- ((u'injuries', u'NNS'), u'amod', (u'grievous', u'JJ'))
- ((u'inflicting', u'VBG'), u'nmod', (u'him', u'PRP'))
- ((u'him', u'PRP'), u'case', (u'on', u'IN'))

Example 1 (Continued)

- Node Metadata
- <Feature_Name.Feature_ Value> pairs
- 1.R.label_crime.name_attacked.result_inflicting grievous injuries on him
- 2.N.label_who.name_An enraged Danniah
- 3.N.label_whom.name_Ashok with an axe

Node Features

- Node Tag
  - R – Relation Node
  - N-Node
- Node Label
  - R label – crime
  - N label – who , whom
- Name
  - R Name – crime name (crime verb)
  - N Name – Name or Description of who / whom

Processing of Dependency Parser Output

- ((u'attacked', u'VBD'), u'nsubj', (u'Danniah', u'NNP'))
  - First entry in nsubj relation
  - 1.R.label_crime.name_attacked.

Processing of Dependency Parser Output

- ((u'attacked', u'VBD'), u'nsubj', (u'Danniah', u'NNP'))
- ((u'Danniah', u'NNP'), u'det', (u'An', u'DT'))
- ((u'Danniah', u'NNP'), u'compound', (u'enraged', u'NNP'))
- ‘Danniah’ as who name – second entry in nsubj relation
- Identify the relations in which ‘Danniah’ occurs and collect the words appearing as the second entry
- Organize the words as they appear in the sentence order
- 2.N.label_who.name_An enraged Danniah

Processing of Dependency Parser Output

- ((u'attacked', u'VBD'), u'dobj', (u'Ashok', u'NNP'))
- ((u'Ashok', u'NNP'), u'nmod', (u'axe', u'NN'))
- ((u'axe', u'NN'), u'case', (u'with', u'IN'))
- ((u'axe', u'NN'), u'det', (u'an', u'DT'))
- ‘Ashok’ as whom name – second entry in dobj relation
- Identify the relations in which ‘Ashok’ occurs and collect the words appearing as the second entry
- Organize the words as they appear in the sentence order
- 2.N.label_whom.name_Ashok with an axe
- Note: Due to error in parsing ((u'Ashok', u'NNP'), u'nmod', (u'axe', u'NN'))
- Should be ((u'attacked', u'VBD'), u'nmod', (u'axe', u'NN'))
- Instrument feature of Crime : an axe is missed

Processing of Dependency Parser Output

- ((u'attacked', u'VBD'), u'advcl', (u'inflicting', u'VBG'))
- ((u'inflicting', u'VBG'), u'dobj', (u'injuries', u'NNS'))
- ((u'injuries', u'NNS'), u'amod', (u'grievous', u'JJ'))
- ((u'inflicting', u'VBG'), u'nmod', (u'him', u'PRP'))
- ((u'him', u'PRP'), u'case', (u'on', u'IN'))
- Relation ‘advcl’ considered to point to ‘result’ feature
- Start with the second entry in the ‘advcl’ relation, consider all the relations in which the word appears, recursively consider all relations in which the second entry of all such relations occur
- Organize the set of words in the sentence order
- 1.R.label_crime.name_attacked.result_inflicting grievous injuries on him
- The processing is over as all the relations have been scanned and processed

Example 2

- A farmer allegedly attacked his wife and her ‘friend’ with an axe at Hayathnagar on Sunday morning.
- ((u'attacked', u'VBD'), u'nsubj', (u'farmer', u'NN'))
- ((u'farmer', u'NN'), u'det', (u'A', u'DT'))
- ((u'attacked', u'VBD'), u'advmod', (u'allegedly', u'RB'))
- ((u'attacked', u'VBD'), u'dobj', (u'wife', u'NN'))
- ((u'wife', u'NN'), u'nmod:poss', (u'his', u'PRP$'))
- ((u'wife', u'NN'), u'cc', (u'and', u'CC'))
- ((u'wife', u'NN'), u'conj', (u'friend', u'NN'))
- ((u'friend', u'NN'), u'nmod:poss', (u'her', u'PRP$'))
- ((u'attacked', u'VBD'), u'nmod', (u'axe', u'NN'))
- ((u'axe', u'NN'), u'case', (u'with', u'IN'))
- ((u'axe', u'NN'), u'det', (u'an', u'DT'))
- ((u'axe', u'NN'), u'nmod', (u'Hayathnagar', u'NNP'))
- ((u'Hayathnagar', u'NNP'), u'case', (u'at', u'IN'))
- ((u'attacked', u'VBD'), u'nmod', (u'Sunday', u'NNP'))
- ((u'Sunday', u'NNP'), u'case', (u'on', u'IN'))
- ((u'attacked', u'VBD'), u'nmod:tmod', (u'morning', u'NN'))

Example 2 (Continued)

- Node Metadata
- <Feature_Name.Feature_ Value> pairs
- 1.R.label_crime.name_attacked.instrument_an axe at Hayathnagar.time_Sunday morning 2.N.label_who.name_A farmer
- 3.N.label_whom.name_his wife and her friend.

Node Features

- Node Tag
  - R – Relation Node
  - N-Node
- Node Label
  - R label – crime
  - N label – who , whom
- Name
  - R Name – crime name (crime verb)
  - N Name – Name or Description of who / whom

Processing of Dependency Parser Output

- ((u'attacked', u'VBD'), u'nsubj', (u'farmer', u'NN'))
  - First entry in nsubj relation
  - 1.R.label_crime.name_attacked.

Processing of Dependency Parser Output

- ((u'attacked', u'VBD'), u'nsubj', (u'farmer', u'NN'))
- ((u'farmer', u'NN'), u'det', (u'A', u'DT'))
- ‘farmer’ as who name – second entry in nsubj relation
- Identify the relations in which ‘farmer’ occurs and collect the words appearing as the second entry
- Organize the words as they appear in the sentence order
- 2.N.label_who.name_A farmer

Processing of Dependency Parser Output

- ((u'attacked', u'VBD'), u'advmod', (u'allegedly', u'RB'))
- ‘advmod’ relation with crime ‘attacked’ describes level of attack
- 1.R.label_crime.name_attacked.level_allegedly

Processing of Dependency Parser Output

- ((u'attacked', u'VBD'), u'dobj', (u'wife', u'NN'))
- ((u'wife', u'NN'), u'nmod:poss', (u'his', u'PRP$'))
- ((u'wife', u'NN'), u'cc', (u'and', u'CC'))
- ((u'wife', u'NN'), u'conj', (u'friend', u'NN'))
- ((u'friend', u'NN'), u'nmod:poss', (u'her', u'PRP$'))
- ‘wife’ as whom name – second entry in dobj relation
- Identify the relations in which ‘wife’ occurs and collect the words appearing as the second entry, recursively consider all relations
- Organize the words as they appear in the sentence order
- 2.N.label_whom.name_his wife and her friend

Processing of Dependency Parser Output

- ((u'attacked', u'VBD'), u'nmod', (u'axe', u'NN'))
((u'axe', u'NN'), u'case', (u'with', u'IN'))
((u'axe', u'NN'), u'det', (u'an', u'DT'))
((u'axe', u'NN'), u'nmod', (u'Hayathnagar', u'NNP'))
((u'Hayathnagar', u'NNP'), u'case', (u'at', u'IN'))
Relation ‘nmod’ considered to point to R node features
Start with the second entry in the ‘nmod’ relation, consider all the relations in which the word appears, recursively consider all relations in which the second entry of all such relations occur
((u'axe', u'NN'), u'case', (u'with', u'IN')) signifies Instrument relation
Organize the set of words in the sentence order
1.R.label_crime.name_attacked.instrument_an axe at Hayathnagar
Wrong relation ((u'axe', u'NN'), u'nmod', (u'Hayathnagar', u'NNP'))
Should be ((u'attacked', u'VBD'), u'nmod', (u'Hayathnagar', u'NNP'))

Processing of Dependency Parser Output

- ((u'attacked', u'VBD'), u'nmod', (u'Sunday', u'NNP'))
- ((u'Sunday', u'NNP'), u'case', (u'on', u'IN'))
- ((u'attacked', u'VBD'), u'nmod:tmod', (u'morning', u'NN'))
- Relation ‘nmod’ considered to point to R node features
- Start with the second entry in the ‘nmod’ relation, consider all the relations in which the word appears, recursively consider all relations in which the second entry of all such relations occur
- nmod:tmod does not start a new feature
- 1.R.label_crime.name_attacked.instrument_an axe at Hayathnagar.time_Sunday morning

Example 3

- A 30-year-old woman was attacked with a cleaning acid allegedly by her uncle in an incident at Katrapadu village in Pedanandipadu mandal.
- ((u'attacked', u'VBN'), u'nsubjpass', (u'woman', u'NN'))
- ((u'woman', u'NN'), u'det', (u'A', u'DT'))
- ((u'woman', u'NN'), u'amod', (u'30-year-old', u'JJ'))
- ((u'attacked', u'VBN'), u'auxpass', (u'was', u'VBD'))
- ((u'attacked', u'VBN'), u'nmod', (u'acid', u'NN'))
- ((u'acid', u'NN'), u'case', (u'with', u'IN'))
- ((u'acid', u'NN'), u'det', (u'a', u'DT'))
- ((u'acid', u'NN'), u'compound', (u'cleaning', u'NN'))
- ((u'attacked', u'VBN'), u'nmod', (u'uncle', u'NN'))
- ((u'uncle', u'NN'), u'advmod', (u'allegedly', u'RB'))
- ((u'uncle', u'NN'), u'case', (u'by', u'IN'))
- ((u'uncle', u'NN'), u'nmod:poss', (u'her', u'PRP$'))
- ((u'uncle', u'NN'), u'nmod', (u'incident', u'NN'))
- ((u'incident', u'NN'), u'case', (u'in', u'IN'))
- ((u'incident', u'NN'), u'det', (u'an', u'DT'))
- ((u'incident', u'NN'), u'nmod', (u'village', u'NN'))
- ((u'village', u'NN'), u'case', (u'at', u'IN'))
- ((u'village', u'NN'), u'compound', (u'Katrapadu', u'NNP'))
- ((u'attacked', u'VBN'), u'nmod', (u'mandal', u'NNP'))
- ((u'mandal', u'NNP'), u'case', (u'in', u'IN'))
- ((u'mandal', u'NNP'), u'compound', (u'Pedanandipadu', u'NNP'))

Example 3 (Continued)

- 1.R.label_crime.name_attacked.instrument_a cleaning acid.location_ Pedanandipadu mandal
- 2.N.label_who.name_A 30-year-old woman
- 3.N.label_who.name_allegedly by her uncle in an incident at Katrapadu village

Processing of Dependency Parser Output

- ((u'attacked', u'VBN'), u'nsubjpass', (u'woman', u'NN'))
  - First entry in nsubjpass relation
  - Sentence is in passive voice
  - 1.R.label_crime.name_attacked.

Processing of Dependency Parser Output

- ((u'attacked', u'VBN'), u'nsubjpass', (u'woman', u'NN'))
- ((u'woman', u'NN'), u'det', (u'A', u'DT'))
- ((u'woman', u'NN'), u'amod', (u'30-year-old', u'JJ'))
- ‘woman’ as whom name – second entry in nsubjpass relation
- Identify the relations in which ‘woman’ occurs and collect the words appearing as the second entry
- Organize the words as they appear in the sentence order
- 2.N.label_who.name_A 30-year-old woman

Processing of Dependency Parser Output

- ((u'attacked', u'VBN'), u'auxpass', (u'was', u'VBD')) → ignore auxpass relation
- ((u'attacked', u'VBN'), u'nmod', (u'acid', u'NN'))
- ((u'acid', u'NN'), u'case', (u'with', u'IN'))
- ((u'acid', u'NN'), u'det', (u'a', u'DT'))
- ((u'acid', u'NN'), u'compound', (u'cleaning', u'NN'))
- Relation ‘nmod’ considered to point to R node features
- Start with the second entry in the ‘nmod’ relation, consider all the relations in which the word appears, recursively consider all relations in which the second entry of all such relations occur
- ((u'acid', u'NN'), u'case', (u'with', u'IN')) signifies Instrument relation
- Organize the set of words in the sentence order
- 1.R.label_crime.name_attacked.instrument_a cleaning acid

Processing of Dependency Parser Output

- ((u'attacked', u'VBN'), u'nmod', (u'uncle', u'NN'))
- ((u'uncle', u'NN'), u'advmod', (u'allegedly', u'RB'))
- ((u'uncle', u'NN'), u'case', (u'by', u'IN'))
- ((u'uncle', u'NN'), u'nmod:poss', (u'her', u'PRP$'))
- ((u'uncle', u'NN'), u'nmod', (u'incident', u'NN'))
- ((u'incident', u'NN'), u'case', (u'in', u'IN'))
- ((u'incident', u'NN'), u'det', (u'an', u'DT'))
- ((u'incident', u'NN'), u'nmod', (u'village', u'NN'))
- ((u'village', u'NN'), u'case', (u'at', u'IN'))
- ((u'village', u'NN'), u'compound', (u'Katrapadu', u'NNP'))
- ‘uncle’ as who name – nmod and case by relation
- Identify the relations in which ‘uncle’ occurs and collect the words appearing as the second entry, recursively consider all relations
- Organize the words as they appear in the sentence order
- 3.N.label_who.name_allegedly by her uncle in an incident at Katrapadu village

Processing of Dependency Parser Output

- ((u'attacked', u'VBN'), u'nmod', (u'mandal', u'NNP'))
- ((u'mandal', u'NNP'), u'case', (u'in', u'IN'))
- ((u'mandal', u'NNP'), u'compound', (u'Pedanandipadu', u'NNP'))
- Relation ‘nmod’ considered to point to R node features
- Start with the second entry in the ‘nmod’ relation, consider all the relations in which the word appears, recursively consider all relations in which the second entry of all such relations occur
- ((u'mandal', u'NNP'), u'case', (u'in', u'IN')) signifies location relation
- Organize the set of words in the sentence order
- 1.R.label_crime.name_attacked.instrument_a cleaning acid.location_ Pedanandipadu mandal
- Processing is over as all the relations have been identified

Node Metadata (9 nodes)

- 1.R.label_crime.name_attacked.result_inflicting grievous injuries on him
- 2.N.label_who.name_An enraged Danniah
- 3.N.label_whom.name_Ashok with an axe
- 4.R.label_crime.name_attacked.instrument_an axe at Hayathnagar.time_Sunday morning 5.N.label_who.name_A farmer
- 6.N.label_whom.name_his wife and her friend.
- 7.R.label_crime.name_attacked.instrument_a cleaning acid.location_ Pedanandipadu mandal
- 8.N.label_who.name_A 30-year-old woman
- 9.N.label_who.name_allegedly by her uncle in an incident at Katrapadu village

Edge List with the 9 nodes

- Edge formation strategy
  - The two N nodes from a sentence form edges with the R node from the same sentence
  - The R nodes are connected across sentences
- Edge List
  - 1 2
  - 1 3
  - 4 5
  - 4 6
  - 7 8
  - 7 9
  - 1 4
  - 1 7
  - 4 7

Clustering Output

- # Input: edgelist.txt
- # Nodes: 9 Edges: 9
- # Algorithm: Clauset-Newman-Moore
- # Modularity: 0.333333
- # Communities: 3
- # NId CommunityId
- 1 0
- 2 0
- 3 0
- 4 1
- 5 1
- 6 1
- 7 2
- 8 2
- 9 2

Incremental Clustering Output

- New edge added ->10 11
- In some sentences the perpetrator information is not present but the victim information is there.
- The newly added nodes are clustered in a separate cluster
- Modularity also increases to 0.420000

Graph Clustering

- Community Structure
  - The gathering of vertices into groups such that there is a high density of edges within groups than between them
- Community Detection
  - Graph Clustering
  - Girvan and Newman
    - uses edge betweenness as a metric to identify the boundaries between communities
    - Time complexity O((m^2)*n) – m edges and n vertices
    - O(n^3) for a sparse graph – m~n

Clauset-Newman-Moore Graph Clustering Algorithm

- Algorithm based on the greedy optimization of modularity
- Time complexity O(mdlogn) – m edges, n vertices, d depth of the dendrogram (network community structure)
- O(nlog^2n) for a sparse graph – m~n, d~logn
- Substantially faster than Girvan and Newman algorithm

Modularity of a Graph

- A property of a graph (network) and a specific proposed division (clustering) of that graph (network) into communities
- It measures when the division is a good one, in the sense that there are many edges within communities and only a few between them

Modularity of a Graph

- Vertices are divided into communities
- Fraction of edges that fall within communities that connect vertices that both lie in the same community
- Subtract from it the expected value of the same quantity in the case of a randomized network
- We get the modularity
- A value above 0.3 is a good indicator of significant community structure in a graph.

Clauset-Newman-Moore Graph Clustering Algorithm

- Greedy optimization technique <bottom-up>
- Starting with each vertex being the sole member of a community of one, two communities are joined repeatedly whose amalgamation increases the modularity metric
- Result is a tree (Dendrogram) whose leaves are the vertices of the original graph and whose internal nodes correspond to the joins.
- Dendrogram – hierarchical decomposition

- The fraction of edges that join vertices in community i to vertices in

$$e_{ij} = \frac{1}{2m} \sum_{vw} A_{vw} \delta(c_v, i) \delta(c_w, j)$$

- The fraction of ends of edges that are attached to vertices in community i.

$$a_i = \frac{1}{2m} \sum_{v} k_v \delta(c_v, i)$$

- Modularity Q can be defined as
  –

$$
\begin{aligned}
Q &= \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \sum_i \delta(c_v, i) \delta(c_w, i) \\
&= \sum_i \left[\frac{1}{2m} \sum_{vw} A_{vw}\, \delta(c_v, i) \delta(c_w, i) \right. \\
&\qquad \left. - \frac{1}{2m} \sum_v k_v\, \delta(c_v, i) \frac{1}{2m} \sum_w k_w \delta(c_w, i) \right] \\
&= \sum_i (e_{ii} - a_i^2).
\end{aligned}
$$

- Three data structures are maintained
- 1. A sparse matrix containing ΔQ_{ij} for each pair i, j of communities with at least one edge between them. Each row of the matrix both as a balanced binary tree (so that elements can be found or inserted in O(log n) time) and as a max-heap (so that the largest element can be found in constant time).
- 2. A max-heap H containing the largest element of each row of the matrix ΔQ_{ij} along with the labels i,j of the corresponding pair of communities.
- 3. An ordinary vector array with elements a_i -> the fraction of ends of edges that are attached to vertices in community i

- initially set

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j/(2m)^2 & \text{if } i, j \text{ are connected,} \\ 0 & \text{otherwise,} \end{cases}$$

- and $a_i = k_i/2m$
- for each i.

Algorithm

- 1. Calculate the initial values of Δ Qij and a_i according to (8) and (9), and populate the max-heap with the largest element of each row of the matrix Δ Q.
- 2. Select the largest Δ Qij from H, join the corresponding communities, update the matrix Δ Q, the heap H and a_i and increment Q by Δ Qij .
- 3. Repeat step 2 until only one community remains.

Future Work

- The word ordering in the metadata can be improved following the tree structure.
- The Stanford dependency parser is giving some incorrect results described above like axe associated with a location instead of verb. This can be solved using sense disambiguation tools like Babel Net.
- Relationships with more verbs need to be investigated for the model.

Reference

- 1. Aaron Clauset, M. E. J. Newman, and Cristopher Moore, Finding Community Structure in Very Large Networks, Physics Review, Volume 70, 2004.

Thank You