	Undirected	Directed
Mean Degree	$\bar{d} = \frac{\sum_{i=1}^{g} d(n_i)}{g} = \frac{2L}{g}.$	$\bar{d}_I = \frac{\sum_{i=1}^{g} d_I(n_i)}{g}$ $\bar{d}_O = \frac{\sum_{i=1}^{g} d_O(n_i)}{g}$ $\bar{d}_I = \bar{d}_O = \frac{L}{g}$
Variance of degree	$S_D^2 = \frac{\sum_{i=1}^{g}(d(n_i) - \bar{d})^2}{g}.$	$S_{D_I}^2 = \frac{\sum_{i=1}^{g}(d_I(n_i) - \bar{d}_I)^2}{g}$ $S_{D_O}^2 = \frac{\sum_{i=1}^{g}(d_O(n_i) - \bar{d}_O)^2}{g}$
Density	$\Delta = \frac{L}{g(g-1)/2} = \frac{2L}{g(g-1)}.$	$\Delta = \frac{L}{g(g-1)}$

BASIC GRAPH PROPERTIES	
Local Clustering Coefficient	$LCC(vi) = \frac{\#pairs\ of\ connected\ neighbours\ of\ vi}{\#pairs\ of\ neighbours\ of\ vi}$

# Pairs of neighbours of vi 	
 d=degree(vi)
	$\binom{d}{2}$

Global Clustering Coefficient/ Transitivity	$GCC = Transitivity$ $= \frac{\#closed\ triples}{\#connected\ triples} = \frac{3 * \#triangles}{3 * \#triangles + \#non\ triangle\ connected\ triples}$
Density	$Density = \frac{\#\text{lines present}}{Max\ possible\ lines}$
Reciprocity	$\text{Reciprocity} = \frac{\#edges\ reciprocated}{\#edges\ in\ graph}$
Sign of a cycle	$Product(Signs\ on\ all\ edges)$ Balanced if sign>0

SEQUENCE OF VERTICES AND EDGES	
Walk	sequence of nodes and lines, starting and ending with nodes, in which each node is incident with the lines following and preceding it in the sequence
Trail	walk in which **all** of the **lines** are **distinct**, though some node(s) may be included more than once
Path	walk in which **all nodes** and **all lines** are **distinct**
venn diagram here	
Closed Walk	A walk that begins and ends at the same node
Cycle	A closed walk of at least three nodes in which **all lines** are **distinct**, and **all nodes except** the **beginning** and **ending** nodes are **distinct**
Tour	A closed walk in which **each line** in the graph is used **at least once**
Eulerian Trail	special closed trails that include **every line exactly once**
Hamiltonian Cycle	Cycle in which **Every node** in the graph is included **exactly once**
venn diagram here	

SPECIAL SUBGRAPHS		Maximal?
Maximal xyz	*If this xyz is not subgraph of a larger xyz*	
Clique	*maximal complete subgraph of three or more nodes.*	√
n-clique	*maximal (if not complete)subgraph in which the largest geodesic distance between any two nodes is no greater than n.* $$\text{Maximal subgraph}$$ $$d(i,j) \leq n \ \forall vi, vj \in V$$ $$diameter(G) \text{ may or may not be } = n$$	√
n-clan	*n-clique in which the geodesic distance between all nodes in the subgraph is no greater than n for paths* ***within*** *the subgraph.* $$d(i,j) \leq n \ \forall vi, vj \in V$$ $$\text{Paths within the subgraph}$$ $$diameter(G) \text{ may or may not be } = n$$	
n-club	*maximal subgraph of diameter n.* $$\text{Maximal subgraph}$$ $$diameter(G) = n$$	√
k-plex	*maximal subgraph containing |V| nodes in which each node is adjacent to no fewer than gs - k nodes in the subgraph* $$\text{Maximal subgraph}$$ $$degree(i) \geq \|V\| - k$$	√
k-core	*subgraph in which each node is adjacent to at least a minimum number k, of the other nodes in the subgraph.* $$degree(i) \geq k$$	×
k-shell	*subgraph which consists of nodes which are part of k core but not part of (k+1) core*	

EGO N/W	
Size G=ego Alter=Direct neighbour	$$Size(G) = \#Alters\ of\ G = degree(G)$$
Redundancy	$$Redundancy(G's\ alter) = \frac{\#(G's\ alter)'s\ alters\ except\ G}{\#alters\ of\ G}$$
Effective Size	$$Effective\ Size(G) = Size(G) - Sum(\ Redundancy\ of\ all\ G's\ alters)$$
Efficiency	$$Efficiency = \frac{Effective\ Size(G)}{Size(G)}$$
Node with highest Efficiency is chosen as EGO	

SIMILARITY INDICES	
Jaccard Similarity N(vi)=Neighbour set of node vi	$$\sigma jaccard(vi, vj) = \frac{\|N(vi) \cap N(vj)\|}{\|N(vi) \cup N(vj)\|}$$
Cosine Similarity	$$\sigma cosine(vi, vj) = \frac{\|N(vi) \cap N(vj)\|}{\sqrt{\|N(vi)\| * \|N(vj)\|}}$$

EVALUATION METRICS -CONFUSION MATRIX BASED

$TP = \#Pairs\ with\ same\ labels\ that\ are\ in\ same\ community$		
$FP = \#Pairs\ with\ different\ labels\ that\ are\ in\ same\ community$		
$FN = \#Pairs\ with\ same\ labels\ that\ are\ in\ different\ communities$		
$TN = \#Pairs\ with\ different\ labels\ that\ are\ in\ different\ communities$		
Accuracy	$Precision = \frac{TP}{TP + FP + TN + FN}$	
Precision	$Precision = \frac{TP}{TP + FP}$	
Recall	$Recall = \frac{TP}{TP + FN}$	
F-measure	$FMeasure = 2 * \frac{Precision * Recall}{Precision + Recall}$	
EVALUATION METRICS-PURITY		
$N = \#Items\ to\ be\ classified$		
$Ci = set\ of\ elements\ in\ ith\ community$		
$Lj = j\ th\ label$		
$Purity = \frac{1}{N}\sum_{i=1}^{k} max_j \lvert Ci \cap Lj \rvert = \frac{\#\ Items\ that\ have\ labels\ same\ as\ majority\ in\ the\ classified\ community}{\#Items\ to\ be\ classified}$		
EVALUATION METRICS-NMI		
$Y = Class\ Labels$		
$C = Cluster\ Labels$		
$Take\ 0 \times \log_2(0) = 0$		
Entropy of class labels	$H(Y) = -\sum_{i=1}^{\lvert Y \rvert} P(Y = i) \log_2 P(Y = i)$	Y=1,2,3..
Entropy of cluster labels	$H(C) = -\sum_{i=1}^{\lvert C \rvert} P(C = i) \log_2 P(C = i)$	C=1,2,3..
Entropy of labels within each cluster	$H(Y \mid C) = \sum_{j=1}^{\lvert C \rvert}\left[P(C = j)\left(-\sum_{i=1}^{\lvert Y \rvert} P(Y = i \mid C = j) \log_2 P(Y = i \mid C = j)\right)\right]$	$Y=1 \mid C=1$ $Y=2 \mid C=1$: $Y=1 \mid C=2$ $Y=2 \mid C=2$
Mutual Information b/w Y and C	$I(Y; C) = H(Y) - H(Y \mid C)$	
Normalised Mutual Information	$NMI(Y, C) = \frac{2 * I(Y; C)}{H(Y) + H(C)}$	
ALITER FORMULA		

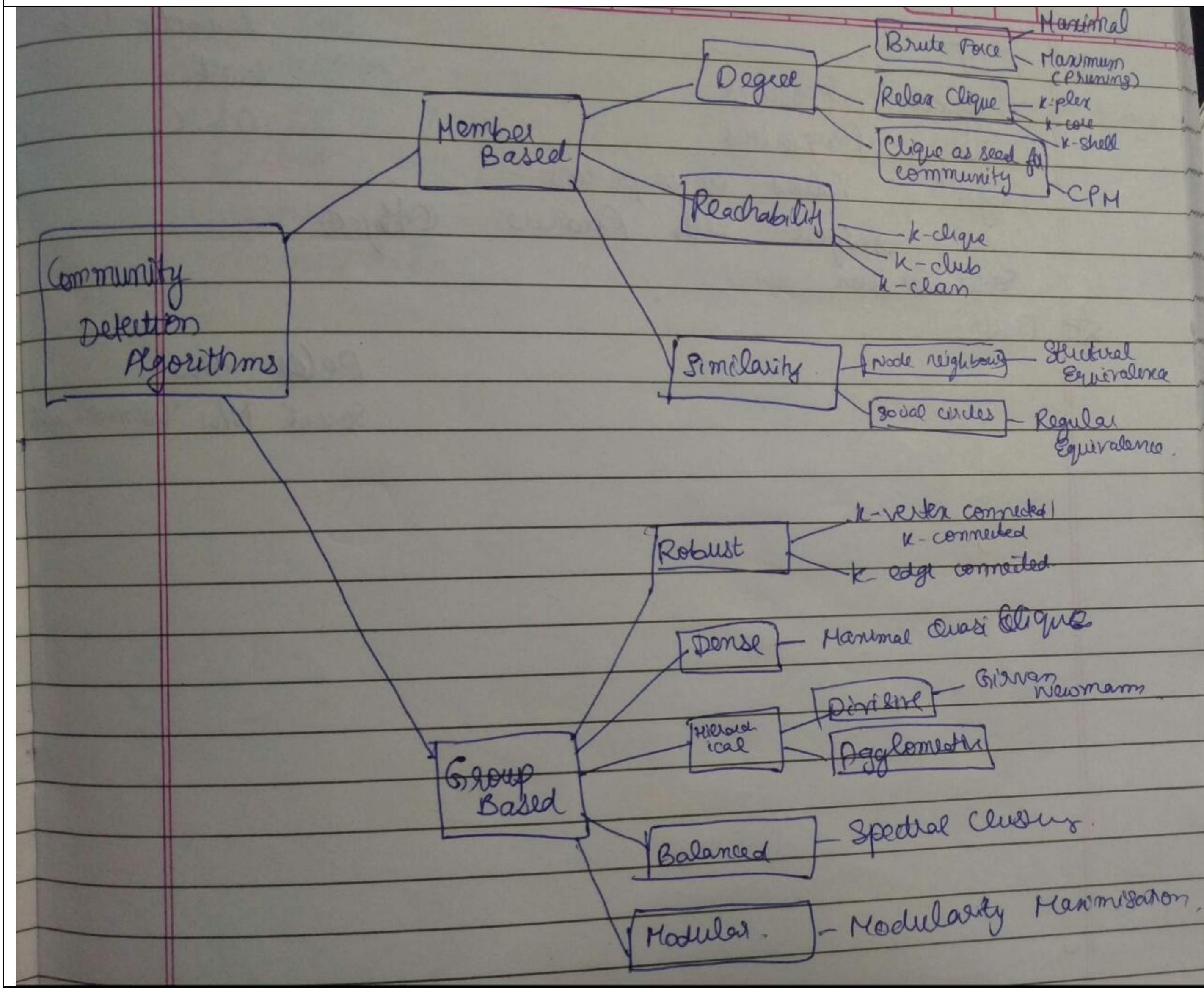
BALANCED COMMUNITIES	
$k = \#Components\ after\ partitioning$	
$\lvert Pi \rvert = \#Vertices\ in\ partition/cutset\ Pi$	
Complement Cut set	$\overline{Pi} = V - Pi$
Size of cut	$cut(Pi, \overline{Pi}) = Sum\ (weights\ of\ edges\ in\ cut)$
Ratio Cut	$Ratio\ Cut(P) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(Pi, \overline{Pi})}{\lvert Pi \rvert}$

Volume of partition	$$vol(Pi) = \sum_{v \in Pi} d(v) = Sum\ (degrees\ of\ vertices\ in\ partition\ Pi)$$ In weighted graph, $Degree(v) = Sum(Weights\ of\ vertices\ that\ meet\ on\ v)$
Normalised Cut	$$Normalised\ Cut(P) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(Pi, \overline{Pi})}{vol(Pi)}$$
Adjacency Matrix	$$Ai,j = \begin{cases} 1 & if\ vi\ adjacent\ to\ vj \\ 0 & otherwise \end{cases}$$
Degree Matrix	$$Di,j = \begin{cases} degree(vi) & if\ i = j \\ 0 & otherwise \end{cases}$$
Laplacian Matrix	$$L = \begin{cases} D - A & Ratio\ Cut\ Laplacian(Unnormalised\ Laplacian) \\ I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} & Normalised\ Cut\ Laplacian(Normalised\ Laplacian) \end{cases}$$
Ratio Cut and Normalised Cut are NP Hard Problems	

SPECTRAL COMMUNITIES	
Adjacency Matrix	$$Ai,j = \begin{cases} 1 & if\ vi\ adjacent\ to\ vj \\ 0 & otherwise \end{cases}$$
Degree Matrix	$$Di,j = \begin{cases} degree(vi) & if\ i = j \\ 0 & otherwise \end{cases}$$
Laplacian Matrix/Admittance Matrix/Kirchoff Matrix/Discrete Laplacian	$$L = D - A$$ $$Li,j = \begin{cases} degree(vi) & if\ i = j \\ -1 & if\ i \neq j\ and\ vi\ is\ adjacent\ to\ vj \\ 0 & otherwise \end{cases}$$
1.Compute the first k eigenvectors of L to get Eigen vector matrix Y 2 Cluster the column Yi into k clusters using the K-means algorithm	

MODULAR COMMUNITIES	
$m = \|E\| = \#Edges\ in\ the\ graph$	
Adjacency Matrix	$$Ai,j = \begin{cases} 1 & if\ vi\ adjacent\ to\ vj \\ 0 & otherwise \end{cases}$$
Degree Matrix	$$Di,j = \begin{cases} degree(vi) & if\ i = j \\ 0 & otherwise \end{cases}$$
Modularity Matrix	$$B = A - \frac{dd^T}{2m}$$ $$Bi,j = Ai,j - \frac{didj}{2m}$$
1) Compute the Eigen values of B 2) Compute top k Eigenvectors of B corresponding to the largest positive eigen values 3) Run k-means on the k Eigenvectors to detect ‘k’ communities	

COMMUNITIES



NETWORK MODELS

	Degree Distribution	Clustering Coeff	Avg Path Length
Real World	Power Law	High	Small
ER Model	Binomial /Poisson	Low	Low
SW Model	Poisson	High	High
BA Model	Power Law	Low	Low

Red => Cause of concern

SMALL WORLD MODEL

1. Degree Distribution

$$P(d_v = d) = \sum_{n=0}^{\min(d-\frac{c}{2},\,\frac{c}{2})} {}^{\frac{c}{2}}C_n \,(1-\beta)^n \,\beta^{\frac{c}{2}-n} \,\frac{\left(\frac{\beta c}{2}\right)^{d-\frac{c}{2}-n}}{d-\frac{c}{2}-n}\, e^{-\frac{\beta c}{2}} \quad \text{(Poisson)}$$

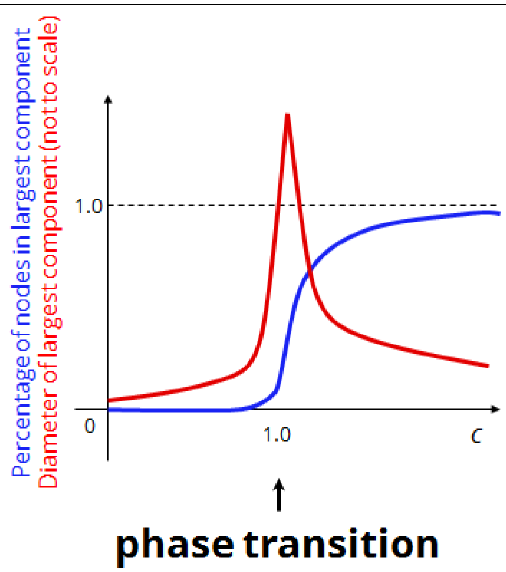
2. Clustering coeff.

$$\frac{3(c-2)}{4(c-1)} \approx \frac{3}{4} \quad \text{(Regular Lattice)}$$

$$C(p) \approx (1-p)^3\, C(0) \quad \text{(small world)}$$

3. Avg Path length : $\frac{n}{2c}$

RANDOM GRAPH NETWORK



If $c < 1$:

- **small**, isolated clusters
- **small** diameters
- **short** path lengths

At $c = 1$:

- a ***giant component*** appears
- diameter **peaks**
- path lengths are **long**

For $c > 1$:

- almost **all** nodes **connected**
- diameter **shrinks**
- path lengths **shorten**

1. # Graphs with n nodes, m edges $= {}^{{}^{n}C_2}C_m$
2. # Edges in $G(n,p)$ expected $= {}^{n}C_2\, p$
3. Avg Degree / Expected Degree $c = (n-1)p$
4. Prob. of observing m edges $= {}^{{}^{n}C_2}C_m\, p^m (1-p)^{{}^{n}C_2 - m}$
5. Prob. that degree of a vertex is d (Degree Distribution) $= {}^{n-1}C_d\, p^d (1-p)^{(n-1)-d}$ (Binomial) $= \frac{e^{-c}c^d}{d!}$ (Poisson)
6. Expected Clustering Coeff: Local $= p$, Global $= p$
7. Avg. Path length $l = \frac{\ln\lvert V\rvert}{\ln c}$
PREFERENTIAL ATTACHMENT MODEL
• **Degree Distribution:** $P(d) = \frac{2m^2}{d^3}$
• **Clustering Coefficient:** $C = \frac{m_0 - 1}{8}\frac{(\ln t)^2}{t}$
• **Average Path Length:** $l \sim \frac{\ln\lvert V\rvert}{\ln(\ln\lvert V\rvert)}$

NAÏVE BAYES THEOREM	
$X = Class\ label$	
$Y = Item\ to\ be\ classified$	
Posterior Probability	$P(X\lvert Y) = \frac{P(X) \times P(Y\lvert X)}{P(Y)}$
NAÏVE BAYES FOR TEXT CLASSIFICATION	
$Ck = Class\ label$	
$Target = Item\ to\ be\ classified$	
$P(Ck\lvert Target) = Given\ a\ Target\ Probability\ that\ it\ belongs\ to\ class\ Ck$	
USING SENTENCES	**USING TFIDF TABLE**

<table>
<tr>
<td>$$P(Ck|Target) = \frac{P(Ck) \times P(Target|Ck)}{P(Target)} = \frac{P(Ck) \times \prod_{w \in Target} P(w|Ck)}{P(Target)}$$
$P(C1|Target) = \frac{P(C1) \times P(Target|C1)}{P(Target)}$,
$P(C2|Target) = \frac{P(C2) \times P(Target|C2)}{P(Target)}$, ...
No need to calculate $P(Target)$</td>
<td>$$P(Ck|Target) = \left(\sum_{w \in Target} TFIDF(w)\right)! \times \prod_{w \in Target} \frac{(P(w|Ck))^{TFIDF(w)}}{TFIDF(w)!}$$</td>
</tr>
<tr>
<td colspan="2">$$P(word = w|Ck) = \frac{\#Instances\ of\ w\ in\ Class\ Ck}{Total\ \#Words(may\ not\ be\ unique)\ classified\ in\ Class\ Ck} = \frac{xk}{d}$$</td>
</tr>
<tr>
<td>If any $P(Target|Ck) == 0$
Apply Laplace Smoothing to all
If $\alpha = 1$, called add-1 smoothing</td>
<td>$$\frac{xk}{d} \rightarrow \frac{xk + \alpha}{N + \alpha d}$$
N=# Unique Words in dataset (including words in Target)</td>
</tr>
<tr>
<td colspan="2">Assign $Target$ to class that has highest $P(Ck|Target)$ value</td>
</tr>
</table>

<table>
<tr><td colspan="2">APRIORI ALGORITHM</td></tr>
<tr><td></td><td>—</td></tr>
</table>

<table>
<tr><td colspan="2">TRUST IN OSN</td></tr>
<tr><td></td><td>—</td></tr>
</table>