TR-2010006: Graph-Based Clustering and Its Application in Coreference Resolution

Zheng Chen

Follow this and additional works at: http://academicworks.cuny.edu/gc_cs_tr

Part of the Computer Sciences Commons

Recommended Citation

This Technical Report is brought to you by CUNY Academic Works. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of CUNY Academic Works. For more information, please contact AcademicWorks@gc.cuny.edu.
Literature Survey:

Graph-based Clustering and its Application in Coreference Resolution

Zheng Chen
zchen1@gc.cuny.edu

Department of Computer Science
The Graduate Center, The City University of New York

Last updated
May 17, 2009
Abstract

In this literature review, we survey graph-based clustering and its application in coreference resolution. We state that the methodology of graph-based clustering can be described by a five-part story: (1) hypothesis which hypothesizes that a graph can be partitioned into densely connected subgraphs that are sparsely connected to each other; (2) modeling which deals with the problem of transforming data into a graph; (3) measure which is an objective function that rates the quality of a clustering; (4) algorithm which aims to optimize the measure; (5) evaluation which evaluates the performance of a system clustering relative to a ground-truth clustering. We then survey coreference resolution which is further split into two problems, entity coreference resolution and event coreference resolution. We focus on discussing how the graph-based clustering methodology has been applied in solving these two problems.
## Contents

Abstract.......................................................................................................................................... i

1. Introduction .......................................................................................................................................... 1

2. Graph-based Clustering Methodology .............................................................................................. 1
   2.1 Graph Notation ................................................................................................................................. 2
   2.2 Hypothesis ......................................................................................................................................... 2
   2.3 Modeling ......................................................................................................................................... 3
   2.4 Measure ........................................................................................................................................... 5
      2.4.1 Measures for a General Hard Clustering Problem ................................................................. 5
      2.4.2 Measures for Graph-based Clustering ...................................................................................... 7
      2.4.3 Summary .................................................................................................................................. 12
   2.5 Algorithm ....................................................................................................................................... 12
      2.5.1 Spectral Clustering Algorithm ............................................................................................... 12
      2.5.2 Markov Clustering Algorithm (MCL) .................................................................................... 16
      2.5.3 Miscellaneous ......................................................................................................................... 17
      2.5.4 Summary .................................................................................................................................. 20
   2.6 Evaluation ...................................................................................................................................... 20
      2.6.1 Formal Constraints on Evaluation Measures ............................................................................ 21
      2.6.2 Evaluation Measures ............................................................................................................... 22
   2.7 Open Problems and Future Directions .......................................................................................... 30

3. Coreference Resolution: an Application in Information Extraction ............................................. 31
   3.1 A Parallel Comparison between Entity Coreference Resolution and Event Coreference Resolution ................................................................................................................................. 32
   3.2 Entity Coreference Resolution ....................................................................................................... 35
   3.3 Event Coreference Resolution ....................................................................................................... 45

4. Conclusions ....................................................................................................................................... 53

References........................................................................................................................................... 53
1. Introduction

In the passing years, there has been a tremendous body of work on graph-based clustering, either done by theoreticians or practitioners. Theoreticians have been busy studying various quality measures (k-median, minimum sum, mincut) and algorithms that exactly or approximately optimize the measures. Unfortunately, evidence has shown that the measures thus far analyzed by theoreticians are easy to fool (Kannan et al., 2000), i.e., “there are simple examples where the right clustering is obvious but optimizing the measures produces undesirable solutions”. Practitioners have been busy working on application-specific algorithms and claiming their effectiveness by taking advantage of the underlying structure or other known characteristics of the data. However, the justifications provided by practitioners are case-by-case and experimental. In this literature survey, we will carry out discussions from theoretical and practical aspects. From the theoretical aspect, we state that the following five-part story describes the general methodology of graph-based clustering:

1. Hypothesis. The hypothesis is that there exist groups of data points in the graph such that the similar data points are assigned in the same group while the dissimilar data points are distinguished by different groups.

2. Modeling. It deals with the problem of transforming data into a graph or modeling the real application as a graph, in which the vertices are data points and edges represent some type of relationship between pairs of data points. There are various forms of constructed graphs, e.g., full connected graph, k-nearest neighbor graph, bipartite graph.

3. Measure. A measure is proposed to answer the question: what exactly is an optimal clustering in the graph?

4. Algorithm. An algorithm is developed to exactly or approximately optimize the quality measure.

5. Evaluation. Evaluation is carried out when a specific algorithm produces a set of clusters. First, humans can look into the clusters and make an intuitive sense of the plausibility in each cluster. Second, some form of “ground truth” can be prepared so that various metrics can be used to measure the performance of clustering.

From the practical aspect, we focus on coreference resolution, which is an important topic in the Information Extraction field. We split coreference resolution into two different but similar problems: entity coreference resolution and event coreference resolution. We review how the methodology of graph-based clustering has been applied in both problems, specifically, how to model the problems using graph structure, what measures and algorithms have been applied, and what are the evaluation results. We also compare graph-based clustering with other approaches proposed for both problems and show that graph-based clustering has achieved state-of-the-art performance for both problems.

2. Graph-based Clustering Methodology

Let $X = \{x_1, ..., x_N\}$ be a set of data points, $S = (s_{ij})_{i,j=1,...,N}$ be the similarity matrix in
which each element indicates the similarity \( s_{ij} \geq 0 \) between two data points \( x_i \) and \( x_j \). The goal of clustering is to divide the data points into several groups such that points in the same group are similar and points in different groups are dissimilar. A graph is a nice way to represent the data. Vertices in the graph represent the data points and the edge weight carries the similarity of two vertices. The clustering problem in graph perspective is then formulated as one of partition into the graph such that the edges in the same group have high weights (which means the points within the group are similar to each other) and the edges between different groups have low weight (which means the points in different groups are dissimilar from each other).

2.1 Graph Notation

A graph is a triple \( G=\langle V,E,W \rangle \) where \( V = \{v_1, ..., v_N\} \) is a set of vertices, \( E \subseteq V \times V \) is a set of edges, and \( W = (w_{ij})_{i,j=1,...,N} \) is called adjacency matrix in which each element indicates a non-negative weight (\( w_{ij} \geq 0 \)) between two vertices \( v_i \) and \( v_j \). If \( W \) is symmetric, i.e., \( w_{ij} = w_{ji} \), \( G \) is called undirected, otherwise, it is directed. The degree of a vertex \( v_i \in V \) is defined as

\[
d_i = \sum_{j=1}^{N} w_{ij}
\]

The degree matrix \( D \) is defined as the diagonal matrix with the degrees \( d_1, ..., d_n \) on the diagonal.

Let \( \mathcal{C} = (C_1, ..., C_K) \) be a partition of \( V \) such that

1. \( C_i \neq \emptyset \) for \( i \in \{1, ..., K\} \).
2. \( C_i \cap C_j = \emptyset \) for \( i,j \in \{1, ..., K\} \) and \( i \neq j \)
3. \( C_1 \cup ... \cup C_K = V \)

We call \( \mathcal{C} \) a clustering of \( G \). \( \mathcal{C} \) is called trivial if either \( K = 1 \), or all clusters \( C_i \) contain only one element.

Each cluster identifies a subgraph of \( G \), i.e., the graph \( G(C_i) = (C_i, E(C_i), W(C_i)) \) where \( E(C_i) = \{(v_m, v_n) \in E: v_m, v_n \in C_i\} \) and \( W(C_i) \) is a submatrix of \( W \) by selecting the rows and columns with index \( m \) for any \( v_m \in C_i \).

2.2 Hypothesis

Although it is in general difficult to answer a question, “what are natural groups in the graph?”, a hypothesis can be made within the methodology of graph-based clustering. The hypothesis can be stated in different ways:

1. There are dense subgraphs (clusters) in the graph such that a dense subgraph contains more and better well-connected internal edges connecting the vertices in the subgraph than cutting edges connecting the vertices across subgraphs.
2. A random walk that visits a dense subgraph will likely stay in the subgraph until many of
its vertices have been visited (Dongen, 2000).

(3) Considering all shortest paths between all pairs of vertices, links between dense subgraphs are likely to be in many shortest paths (Dongen, 2000).

The above three statements are in fact strongly connected to each other and the situation can be explained by an example of social graph. In one kind of social graph, researchers are connected to each other by common research interests and researchers are viewed as vertices in the graph. Obviously, we have clusters (a better word in this example is community) characterized by the research interests and the above three statements can be interpreted as:

(1) Researchers in the same community have more interactions among themselves than outside community, e.g., meeting each other in the conference, collaborating in a project, coauthoring a paper.

(2) A researcher will likely read papers from others in the same community if he or she randomly read papers. Alternatively, a researcher will likely surf the web pages of other researchers in the same community if he or she randomly surf the web pages of all researchers.

(3) Those researchers that have connections in multi-communities will likely enhance cooperation among communities and introduce researchers from different communities to know each other.

Once we make the hypothesis, the key issue now is to measure and quantify the intra-cluster density, inter-cluster sparsity, random walk and shortest path. We will state this issue in the next section.

2.3 Modeling

When we model a set of data points as a graph, it is intuitive that the vertices on the graph represent the data points and the edge weights represent the similarities between pairs of data points. However, when we model Information Extraction problems as graphs, the meaning of vertices and edge weights can be varied from case to case, for example, for entity coreference resolution problem, the vertices on the graph represent entity mentions and the edges carry the coreference relationship between entity mentions.

Besides making clear the meaning of vertices and edges in the graph, a more critical issue is the computation of similarities or distances between two vertices before we start constructing the graph. The choice of such similarity computation also varies from case to case and depends on the application the data comes from. But basically, we need to make sure that the computed values are meaningful which means that two vertices with a high similarity score are indeed closely related in the real application. For example, when designing a similarity function for entity coreference resolution, it makes sense to check whether two entity mentions with a high similarity (coreference) score do corefer to each other.

There are several popular graph construction methods as stated in (Luxburg, 2006):

- **The ε-neighborhood graph**: the graph is constructed by connecting vertices whose pairwise distances are smaller than ε. Correspondingly we have a δ-neighborhood graph which is constructed by connecting vertices whose pairwise similarities are greater than δ. Usually we consider these two graphs as unweighted graphs because the distances between all connected vertices are roughly of the same scale (at most ε.
or at least $\delta$.

- **$k$-nearest neighbor graph**: the graph is constructed by connecting vertex $v_i$ and $v_j$ if $v_i$ is among the $k$-nearest neighbors of $v_j$ or if $v_j$ is among the $k$-nearest neighbors of $v_i$. An alternative is to connect $v_i$ and $v_j$ if both $v_i$ and $v_j$ are among the $k$-nearest neighbors of the other. The resulting graph is called **mutual $k$-nearest neighbor graph**. For both graphs, we weight the edges by the similarity of their end points.

- **The fully connected graph**: the graph is constructed by simply connecting all vertices with positive similarity with each other, and we weight all edges by the similarity of their end points.

Although all graphs mentioned above are generally used, there are two key points that should attract our attentions: (1) which graph should be used and how does it affect the clustering algorithm? (2) how to choose the parameter ($\epsilon, \delta$, or $k$) and how does it affect the clustering algorithm? Empirical experiments show that some clustering algorithm (e.g., spectral clustering) can be quite sensitive to the choice of graphs and parameters. Unfortunately, systematic theoretical study on the above two points do not exist.

Luxburg (2006) studied the behavior of the different graphs by a toy example shown in Figure 1 which shows three clusters: two “moons” (the bottom one is denser than the top one) and a Gaussian. We summarize Luxburg’s results as follows:

- **$\epsilon$-neighborhood graph**: it tends to connect points within regions of high density while disconnect points within regions of low density.

- **$k$-nearest neighbor graph**: besides connecting points within regions of high density, it also connects points in regions of different densities (e.g., one point from low-density Gaussian, and the second point from high-density bottom moon).

- **mutual $k$-nearest neighbor graph**: it tends to connect points within regions of constant density, but does not connect regions of different densities with each other.

Luxburg (2006) also analyzed some potential effects of those graphs on the spectral clustering. As he pointed out, if the graph contains more connected components than the number of clusters we ask the algorithm to detect, spectral clustering will trivially return connected components as clusters. Therefore, unless one is quite sure the connected components are the correct clusters, one should make sure that the graph is “safely” connected, in other words, the graph contains very few or no isolated vertices. Despite his insight into this problem, he admitted that the theoretical results of how the graph could be “safely” connected (which graph and what parameter) are barely known. Furthermore, he limited his discussions in the spectral clustering; therefore his arguments may not always stand in the other graph-based clustering algorithms.

In sum, although graph construction is a basic problem in the methodology of graph-based clustering, it is not well studied and theoretical justifications around the two points (which graph and parameter choice) should be considered as interesting and important topics for future research.
2.4 Measure

A measure quantitatively answers the question: what exactly is an optimal clustering? In the clustering literature, researchers have made great efforts to propose various measures, thus as they claimed, they achieved “optimal” clustering according to the specified measure. In this section, we will first briefly describe some long-standing measures in the general area of clustering and then focus on the measures that capture the characteristics of graphs in the methodology of graph-based clustering.

2.4.1 Measures for a General Hard Clustering Problem

A general hard clustering problem is that given a set of $N$ points $C$, we seek the optimal partition of $C$ into $K$ subsets, $C_1, \ldots, C_K$, $C_i \cap C_j = \emptyset$ for any $i \neq j$ and $C_1 \cup \ldots \cup C_K = C$.

Some well-known measures as listed as follows:

(1) Minimum diameter (Charikar et al., 1997)

The diameter of a cluster is defined to be the maximum inter-point distance in it. The objective is to minimize the maximum cluster diameter, i.e.,

$$\min \max_{1 \leq i \leq K} \text{diameter}(C_i)$$

where $\text{diameter}(C_i) = \max \{|x_i, x_j| \mid x_i, x_j \in C_i\}$

(2) K-median (Charikar et al., 1999)

This measure involves selecting at most $K$ data points as cluster centers, and assigning each data point $j$ to a center $i$ with cost $w_{ij}$. The objective is to minimize the sum of the assignment costs, i.e.,
\begin{align*}
\text{minimize} & \quad \sum_{i,j \in N} w_{ij} x_{ij} \tag{4} \\
\text{subject to} & \quad \sum_{i \in N} x_{ij} = 1, \text{ for each } j \in N, \tag{5} \\
& \quad x_{ij} \leq y_{i}, \text{ for each } i, j \in N \tag{6} \\
& \quad \sum_{i \in N} y_{i} \leq K, \tag{7} \\
& \quad x_{ij} \in \{0,1\}, \text{ for each } i, j \in N \tag{8} \\
& \quad y_{i} \in \{0,1\}, \text{ for each } i \in N \tag{9}
\end{align*}

The constraint of (5) ensures that each data point \( j \in N \) is assigned to some center \( i \in N \), (6) ensures that no data point \( j \in N \) will be assigned to a non-center \( i \in N \), (7) ensures that there are at most \( k \) centers.

(3) Minimum sum (Indyk, 1999)

The objective is to minimize the sum of inter-point distance in all clusters, i.e.,

\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{K} \sum_{x_{i} \in C_{i}, x_{j} \in C_{i}} d(x_{i}, x_{j}) \tag{10}
\end{align*}

Although the above measures are mathematically attractive and simple, they are easy to fool. Kannan et al. (2000) illustrated that optimizing the above measures may in fact produce obvious “bad” clusters. Two examples are shown in Figure 2 (a) and (b) respectively. In Figure 2 (a), although clustering A leads to a larger maximum diameter, it is more desirable than B. The problem also arises for the minimum sum (Figure 2.a) and K-median (Figure 2.b) measure.

Figure 2 (a) Optimizing minimum diameter or minimum sum produces clustering B but A is more desirable. (b) Optimizing K-median produces clustering B but A is more desirable. (Kannan et al., 2000)
2.4.2 Measures for Graph-based Clustering

Following the notation given in section 2.1, graph clustering is to identifying sparsely connected dense subgraphs (clusters) in a given graph and the goal can be achieved by optimizing a fitness function (measure) that measures the quality of clustering within the graph. We review the measures as follows:

1. intra-cluster density
   It is measured by the fraction of the sum of edge weights inside a cluster with respect to the sum of weights in the graph, i.e.,
   \[
   \text{intra}_\text{density}(C_i) = \frac{\sum_{u \in C_i, v \in C_i} w_{uv}}{\sum_{(u, v) \in E} w_{uv}}
   \]
   where \( C_i \) is a cluster (subgraph) in the graph, \( w(u, v) \) is the weight of edge \((u, v)\), \( E \) is the set of edges in the graph. If the graph is unweighted, intra-cluster density equals the number of edges in the subgraph.
   The objective is to maximize the sum of intra-cluster density for all clusters, i.e.,
   \[
   \max \sum_{i=1}^{K} \text{intra}_\text{density}(C_i)
   \] (12)

2. inter-cluster density
   It is measured by the fraction of the sum of edge weights across two subgraphs with respect to the sum of weights in the graph, i.e.,
   \[
   \text{inter}_\text{density}(C_i, C_j) = \frac{\sum_{u \in C_i, v \in C_j} w_{uv}}{\sum_{(u, v) \in E} w_{uv}}
   \]
   where \( C_i, C_j \) are two clusters in the graph, \( w(u, v) \) is the weight of edge \((u, v)\), \( E \) is the set of edges in the graph. If the graph is partitioned into 2 clusters, then the clustering \( C = (S, V \setminus S) \) is called a cut of the graph where \( V \) is the set of vertices and \( S \subset V \). The value of the cut is the sum of the edge weights across two subgraphs, i.e.,
   \[
   \text{cut}(S, V \setminus S) = \sum_{u \in S, v \in V \setminus S} w_{uv}
   \]
   The objective is to minimize the sum of inter-cluster density among clusters, i.e.,
   \[
   \min \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \text{inter}_\text{density}(C_i, C_j)
   \] (15)
   A cut that satisfies (15) is called a mincut.

In fact, (12) and (15) are equivalent to each other, in other words, maximizing the sum of intra-cluster density results in minimizing the sum of inter-cluster sparsity and vice versa. We have the equation as follows:

\[
\sum_{i=1}^{K} \text{intra}_\text{density}(C_i) + \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \text{inter}_\text{sparsity}(C_i, C_j) = 1
\] (16)

The above two measures can give the right clustering in the examples of Figure 2. To see this, we assume that the points in the examples induce an unweighted graph in which two vertices are connected by an edge if they are close together. Clustering (A)
will be obtained in each example. Unfortunately, these two measures favor clusters containing isolated vertices. For an example shown in Figure 3, the two measures produce a less desirable clustering (B) even although B has a smaller mincut.

![Figure 3. Optimizing inter-cluster sparsity produces B but A is more desirable](image)

To achieve a better balance in the cardinality of either side of the cut, it is suggested to optimize the ratio cut (Hagan and Kahng, 1992) or normalized cut (Shi and Malik, 2000).

(3) **ratio cut** (Hagan and Kahng, 1992)
It is defined as:

\[
\text{ratio cut}(C_i, V \setminus C_i) = \frac{\text{cut}(C_i, V \setminus C_i)}{|E(C_i)|}
\]

(17)

where \(|E(C_i)|\) is the number of edges in cluster \(C_i\).

The objective is to minimize the sum of \text{ratio cut} for all clusters, i.e.,

\[
\min \sum_{i=1}^{K} \text{ratio cut}(C_i, V \setminus C_i)
\]

(18)

Ratiocut is suitable for unweighted graph, however, for weighted graph, the number of vertices in a cluster may not correspond to a high intra-cluster density. Therefore, we recommend normalized cut.

(4) **normalized cut** (Shi and Malik, 2000)
Similar to ratio cut, normalized cut is given as:

\[
\text{ncut}(C_i, V \setminus C_i) = \frac{\text{cut}(C_i, V \setminus C_i)}{\text{vol}(C_i)}
\]

(19)

where \(\text{vol}(C_i) = \sum_{u \in C_i} \sum_{v \in V} w_{uv}\)

(20)

The objective is to minimize the sum of \text{ncut} for all clusters, i.e.,

\[
\min \sum_{i=1}^{K} \text{ncut}(C_i, V \setminus C_i)
\]

(21)

The common drawback of ratio cut and normalized cut, in contrast to inter-cluster sparsity, is that it favors clusters with equal size.

(5) **performance** (Brandes et al., 2003)
Performance for an unweighted graph is defined as the fraction of intra-cluster edges
together with non-adjacent pairs of nodes in different clusters within the set of all pairs of nodes. The function \( f \) counts the number of edges within all clusters and the function \( g \) counts the number of non-adjacent pairs belonging to different clusters

\[
f(C) = \sum_{i=1}^{K} |E(C_i)|
\]

\[
g(C) = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} |\{(u, v) \in E | u \in C_i, v \in C_j\}|
\]

performance_{unweighted}(C) = \frac{f(C) + g(C)}{1/2 \ n(n - 1)} \tag{24}

For weighted graph, since weights are unknown for node pairs that are not connected with edges, a meaningful upper bound of the weights \( M \) can be defined. Therefore,

\[
f(C) = \sum_{i=1}^{K} \sum_{u \in C_i, v \in C_i} w_{uv}
\]

\[
g(C) = M \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} |\{(u, v) \in E | u \in C_i, v \in C_j\}|
\]

performance_{weighted}(C) = \frac{f(C) + g(C)}{1/2 \ n(n - 1)M} \tag{27}

The above idea is to assume that non-existing edges all have maximum weights \( M \). Alternatively, we can take the weights of the inter-cluster edges into consideration, and modify \( g(C) \) as follows:

\[
g'(C) = M \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} |\{(u, v) \in E | u \in C_i, v \in C_j\}| + \nu(M \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} |\{(u, v) \in E | u \in C_i, v \in C_j\}| - \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \sum_{u \in C_i, v \in C_j} w_{uv})
\]

where \( \nu \in [0, 1] \) is a scaling parameter.

performance'_{weighted}(C) = \frac{f(C) + g'(C)}{1/2 \ n(n - 1)M} \tag{29}

The objective is to maximize the performance, i.e.,

\[
\text{maximize performance}(C)
\]

(6) expansion (Kannan et al., 2000)

Expansion is a measure of denoting the quality of a cluster and computed by the following formula:

\[
\text{expansion}(C_i) = \min_{(S, C_i \setminus S)} \frac{\sum_{u \in S, v \in C_i \setminus S} w_{uv}}{\min(|S|, |C_i \setminus S|)} \tag{31}
\]
where $C_i$ is a cluster (subgraph) in the graph, $w_{uv}$ is the weight of edge $(u, v)$, and $(S, C_i \setminus S)$ is a cut within $C_i$.

If a cluster has a small expansion, it might suggest that there is a cut that can divide the cluster into two finer pieces which further implies that the cluster itself contains lots of dissimilar vertices and it is of low quality.

The expansion of a clustering is the minimum expansion of one of the clusters.

The objective is to maximize the expansion of a clustering, i.e.,

$$\text{maximize } \min_{1 \leq i \leq K} \text{expansion}(C_i)$$  \hspace{1cm} (32)

(7) conductance (Kannan et al., 2000)

Conductance is similar to expansion except that it weights cuts inversely by a function of edge weight instead of the number of vertices in a cut set, i.e.,

$$\text{conductance } (C_i) = \frac{\sum_{u \in S, v \in C_i \setminus S} w_{uv}}{\min(\text{vol}(S), \text{vol}(C_i \setminus S))}$$  \hspace{1cm} (33)

where $\text{vol}(S) = \sum_{u \in S} \sum_{v \in V} w_{uv}$, $V$ is the set of vertices in the graph.

Similarly, the conductance of a clustering is the minimum conductance of one of the clusters.

The objective is to maximize the conductance of a clustering, i.e.,

$$\text{maximize } \min_{1 \leq i \leq K} \text{conductance } (C_i)$$  \hspace{1cm} (34)

The main difference between expansion and conductance is that expansion treats all vertices as equally important while conductance gives more importance to vertices with high degrees and edge weights.

Both expansion and conductance can give the right clustering in the examples of Figure 2 and Figure 3. Therefore, in general, they are better than the measure of intra-cluster density or inter-cluster sparsity. However, there is still a problem with them because both only impose qualities within the clusters and neither enforces qualities pertaining to inter-cluster weights. Kannan et al. (2000) proposed the following bicriteria to optimize both intra-cluster and inter-cluster qualities.

(8) bicriteria (Kannan et al., 2000)

The bicriteria optimization problem requires: 1) clusters must have some minimum conductance $\alpha$; 2) the total weight of inter-cluster edges is at most an $\varepsilon$ fraction of the total edge weight.

The objective is that given $\alpha$, find a clustering that minimizes $\varepsilon$ or given $\varepsilon$, find a clustering that maximizes $\alpha$.

(9) modularity (Girvan and Newman, 2002)

The modularity proposed by Newman and Girvan is a measure of the quality of a particular cluster and is defined as follows.

$$\text{modularity}(C_i) = e_i - a_i^2, \text{with } a_i = \sum_{j=1}^{K} e_{ij}$$  \hspace{1cm} (35)

where $e_i$ is the fraction of internal edges within the cluster $C_i$, $e_{ij}$ ($j \neq i$) is the fraction of edges that connect vertices from cluster $i$ to $j$, i.e., the probability that a randomly drawn link connects a vertex from cluster $i$ to $j$. $a_i$ represents the fraction of edges that connect to vertices in cluster $i$, i.e., the probability that an edge has an end in
cluster $i$. Therefore, the expected fraction of edges with both ends in cluster $i$ is $a_i^2$.

The modularity can thus be interpreted as: the actual edge density in the cluster $C_i$ minus the expected value within $C_i$ when all vertices in the graph are randomly connected keeping the degree of the vertices fixed.

The modularity of a clustering in the graph is defined as the sum of the modularity of each cluster, i.e.,

$$Q = \sum_{i=1}^{K} \text{modularity}(C_i)$$  \hspace{1cm} (36)

The objective is to maximize the modularity of a clustering, i.e.,

$$\text{maximize } Q$$  \hspace{1cm} (37)

If for each cluster, the actual edge density is no better than random, we will get $Q = 0$. Values approaching $Q = 1$, which is the maximum, indicate the strong clustering structure in the graph. The practical value of $Q$ falls in the range from about 0.3 to 0.7 (Girvan and Newman, 2002).

A variant of $Q$ can be written as follows (Fortunato and Barthelemy, 2007)

$$Q = \sum_{i=1}^{K} \left( \frac{l_i}{L} - \left( \frac{d_i}{2L} \right)^2 \right)$$  \hspace{1cm} (38)

where $l_i$ is the number of edges in the cluster $C_i$, $L$ is the total number of edges in the graph, $d_i$ is the total degree of the vertices in the cluster $C_i$.

Recently, the modularity measure has gained great interests from researchers and has been showed effectiveness in various applications. Unfortunately, modularity also suffers from some drawbacks (Chen et al., 2009).

(i) The modularity requires global knowledge of the graph’s topology, i.e., the number of edges $L$, which is problematic for large and dynamic network such as the World Wide Web. Clauset (2005) proposed a local modularity for graphs in which we do not need to know the global knowledge.

(ii) Fortunato and Barthelemy (2007) proved that the modularity leads to a resolution limit problem which fails to identify clusters smaller than a certain scale. To solve this problem, Ruan and Zhang (2008) proposed a recursive algorithm, HQcut, which detects clusters with a high resolution.

(iii) Scripps et al. (2007) showed that the modularity only measures existing edges in the graph but does not explicitly take non-edges into consideration. In the example shown in Figure 4, the two graphs both have the same number of edges and both have the same $Q$ value of 0.216, but the second has more disconnected vertex pairs within community (cluster) $1$ and clearly is worse than the first one. Therefore, modularity fails to distinguish good from bad clustering between different graphs. To alleviate this problem, Scripps et al. (2007) proposed two ratios $p$ and $q$ to measure the fraction of edges within clusters and absent edges between clusters. Unfortunately, it still fails to present clear interpretation of the quality of different clustering structures in some cases, e.g., one clustering has higher $p$ but lower $q$ than the other. As a step further, Chen et al. (2009) proposed Max-Min Modularity which makes it possible to compare the clustering
structure quality between different graphs.

Figure 4. Two graph examples with same modularity score, but the right has more absent edges than the left graph (Chen et al., 2009)

2.4.3 Summary

Optimizing each of the measures mentioned in section 2.4.2 has been shown as NP-hard problem, specifically, intra-cluster density and inter-cluster sparsity in (Ausiello et al., 2002; Wagner and Wagner, 1993), ncut (Shi and Malik, 2000), expansion and conductance (Ausiello et al., 2002; Šíma and Schaeffer, 2006), bicriteria in (Kannan et al., 2000), performance (Shamir et al., 2002), modularity (Brandes, 2006). As a result, any efficient algorithm, which has been claimed to solve the optimal problem with polynomial-time complexity, is heuristic and yields sub-optimal clustering. We will review some of the well-known algorithms in the next section.

2.5 Algorithm

2.5.1 Spectral Clustering Algorithm

In the literature of graph-based clustering algorithms, spectral clustering absolutely plays an important role, for one reason, it has elegant linear algebra foundation, for another reason, it is simple to implement, can be solved efficiently, and very often outperforms traditional clustering algorithms such as $k$-means algorithm (Luxburg, 2006). The main tools for spectral clustering are graph Laplacian matrices. Following the notations in section 2.1, $W$ represents the adjacency matrix, $D$ represents the degree matrix, and we have:

- Unnormalized graph Laplacian matrix
  \[
  L = D - W
  \]

- Normalized graph Laplacian matrices
  \[
  L_{sym} = D^{-1/2}LD^{-1/2} \\
  L_{rw} = D^{-1}L
  \]

For a complete overview of those matrices’ properties, we refer readers to (Mohar,1991; Mohar, 1997; Chung, 1997). However, some properties related with eigenvalues and eigenvectors of those matrices play key role in the spectral clustering algorithms. We list them as follows:

- 0 is the smallest eigenvalue of $L$ with constant one vector $\mathbb{1}$ as eigenvector. The
same applies to $L_{rw}$, but for $L_{sym}$, 0 is the smallest eigenvalue and $D^{1/2}$ is the eigenvector.

- $L$, $L_{sym}$ and $L_{rw}$ are positive semi-definite and have $n$ non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.
- $\lambda$ is an eigenvalue of $L_{rw}$ with eigenvector $u$ if and only if $\lambda$ and $u$ solve the generalized eigenproblem $Lu = \lambda Du$.

Based on the above three graph Laplacian matrices, we have three corresponding spectral clustering algorithms which are summarized in Table 1, 2, and 3 (Luxburg, 2006). It is worth noting that in Table 2, we use unnormalized Laplacian $L$, but since the algorithm solves the generalized eigenproblem $Lu = \lambda Du$, according to the property mentioned above, it actually works with eigenvectors of $L_{rw}$.

We summarize some key points about the three spectral clustering algorithms as follows, and except the second one, the others were stated in (Luxburg, 2006) and with some extended discussions by us.

- Finding an approximate solution for optimizing ratiocut measure leads to unnormalized spectral clustering (Luxburg, 2006) while optimizing ncut measure leads to normalized spectral clustering (Shi and Malik, 2000).
- The number of clusters $k$ doesn’t need to be pre-defined. Instead, it can be determined by some criterion. For example, we can set up a threshold $q$ such that $k$ is the minimum value that satisfies $\frac{\|U_k\|_F}{\|U\|_F} \geq q$, where $\|U_k\|_F$ represents the Frobenius norm of matrix $U$, i.e., $\|U_k\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^k u_{ij}^2}$, $\|U\|_F$ is similarly defined. The larger the value of $q$, the more clusters we will obtain.
- Which spectral clustering algorithm do we choose? It depends on the degree distribution of the constructed graph. If the graph is regular and most vertices have approximately the same degree, then all the Laplacians are very similar to each other, and will work equally well for clustering. However, if the degrees in the graph are broadly distributed, then we prefer normalized rather than unnormalized spectral clustering (recall that unnormalized computes ratiocut in which the number of vertices may not imply high intra-cluster density while normalized computes ncut in which the sum of the weights in a cluster does imply), and in the normalized case, we prefer $L_{rw}$ rather than $L_{sym}$ (the eigenvectors of $L_{rw}$ are cluster indicator vectors while the eigenvectors of $L_{sym}$ are additionally multiplied by $D^{1/2}$, which might lead to undesired artifacts).
- The success of spectral clustering is mainly based on the fact that it does not make strong assumptions on the form of the clusters. Unlike the $k$-means algorithm, where the resulting clusters form convex sets, spectral clustering can solve very general problems like intertwined spirals.
- Solving a standard eigenvalue problem for all eigenvectors takes $O(n^3)$ where $n$ is the number of vertices in the graph. However, if the graph is sparse, we have a very efficient algorithm named Lanczos algorithm which takes $O(mn)$ where $m$ is the number of steps Lanczos takes to converge, $n$ is the number of vertices. Therefore spectral clustering could be efficient for large data sets, as long as we make sure the
sparsity of the graph which is frequently valid for real applications.

- The algorithm for solving optimization problem often suffers from “local” optimum traps. However, we do not need to worry about that for spectral clustering.
- As mentioned in section 2.2, spectral clustering could be very unstable under different choices of the parameters when constructing the graph. Therefore it is not a panacea, but with some care it is a powerful tool that can produce good results.

Input: a constructed graph with the adjacency matrix $W \in \mathbb{R}^{n \times n}$, degree matrix $D$; number $k$ of clusters for output

- Compute the unnormalized Laplacian $L = D - W$
- Compute the first $k$ eigenvectors $u_1, \ldots, u_k$ of $L$ corresponding to the $k$ smallest eigenvalues
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the column vectors $u_1, \ldots, u_k$
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$-th row of $U$
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the $k$-means algorithm into clusters $Y_1, \ldots, Y_k$.

Output: clusters $C_1, \ldots, C_k$ with $C_i = \{j | y_j \in Y_i\}$

Table 1. Unnormalized spectral clustering algorithm

Input: a constructed graph with the adjacency matrix $W \in \mathbb{R}^{n \times n}$, degree matrix $D$; number $k$ of clusters for output

- Compute the normalized Laplacian $L_{sym} = D^{-1/2} L D^{-1/2}$
- Compute the first $k$ eigenvectors $u_1, \ldots, u_k$ of $L_{sym}$
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the column vectors $u_1, \ldots, u_k$
- Form the matrix $T \in \mathbb{R}^{n \times k}$ from $U$ by normalizing the rows to norm 1, i.e.,
  \[ t_{ij} = \frac{u_{ij}}{\sqrt{\sum_{k=1}^{n} u_{ik}^2}} \]
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$-th row of $T$
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the $k$-means algorithm into clusters $Y_1, \ldots, Y_k$.

Output: clusters $C_1, \ldots, C_k$ with $C_i = \{j | y_j \in Y_i\}$

Table 2. Normalized spectral clustering algorithm I (Shi and Malik, 2000)

Input: a constructed graph with the adjacency matrix $W \in \mathbb{R}^{n \times n}$, degree matrix $D$; number $k$ of clusters for output

- Compute the normalized Laplacian $L_{sym} = D^{-1/2} L D^{-1/2}$
- Compute the first $k$ eigenvectors $u_1, \ldots, u_k$ of $L_{sym}$
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the column vectors $u_1, \ldots, u_k$
- Form the matrix $T \in \mathbb{R}^{n \times k}$ from $U$ by normalizing the rows to norm 1, i.e.,
  \[ t_{ij} = \sqrt{\frac{u_{ij}}{\sum_{k=1}^{n} u_{ik}^2}} \]
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$-th row of $T$
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the $k$-means algorithm into clusters $Y_1, \ldots, Y_k$.

Output: clusters $C_1, \ldots, C_k$ with $C_i = \{j | y_j \in Y_i\}$

Table 3. Normalized spectral clustering algorithm II (Ng, 2002)
Besides the above three spectral clustering algorithms, there are other variations that work on optimizing different quality measures. They are listed as follows:

- Kannan et al. (2000) proposed an *Iterative Conductance Cutting* algorithm which iteratively splits clusters using minimum conductance cuts. They also strictly proved that the approximation algorithm has reasonable worst-case guarantees with respect to the *bicriteria* measure. However, they did not describe great details of the algorithm. Brandes et al. (2003) provided a detailed implementation of the algorithm but with some confusing notations. We would like to provide a little more readable one as shown in Table 4.

**Input:** a constructed graph $G = (V, E)$ with the adjacency matrix $W \in \mathbb{R}^{n \times n}$, degree matrix $D$, conductance threshold $\alpha$

Initially the clustering $C = \{V\}$

while there is a $C \in C$ with the conductance of $G(C)$ less than $\alpha$ do

- Compute the normalized Laplacian of $G(C)$

$$M(G(C)) = D(G(C))^{-1}W(G(C))$$

- Obtain the eigenvector $X = (x_1, ..., x_n)$ corresponding to the second largest eigenvalue of $M(G(C))$

- Form a set of cuts $T = \{(S, C \setminus S)\mid S \subseteq C, \max_{w \in S} \{x_w\} < \min_{w \in C \setminus S} \{x_w\}\}$

- Find a cut in $S$ with the minimum conductance, i.e.,

$$\left(S, C \setminus S\right) = \arg\min_{t=(S, C \setminus S) \in T} \text{conductance}(t)$$

- $C = (C \setminus \{C\}) \cup \{S, C \setminus S\}$

Table 4. Iterative Conductance Cutting (Kannan et al., 2000)

The idea is that for each subgraph $G(C)$ (initially the whole graph $G(V)$), we compute the normalized Laplacian of $G(C)$ (note that it is different from the previously defined normalized Laplacian) and obtain the second eigenvector (corresponding to the second largest eigenvalue rather than smallest); order the values in the vector, and form a set of cuts by splitting the order into two parts such that the maximum value in the first part is less than the minimum value in the second part; then we find a split that obtains the minimum conductance; recurse the procedure on the split set $S$ and $C \setminus S$ until conductance of $S$ or $C \setminus S$ exceeds the input threshold $\alpha$.

The complexity of this algorithm is dominated by the eigenvector computation which is $O(mn)$ by applying Lanczos algorithm.

- Brandes et al. (2003) proposed a *Geometric MST Clustering* algorithm that combines spectral partitioning with a geometric clustering technique. The algorithm is presented in Table 5.

The idea is that we obtain the largest $d'$ eigenvectors with eigenvalues greater than 0, recompute the weights by a distance function; find the minimum spanning tree on $G$ which implies a sequence of clusterings as follows: for a threshold value $\tau$, let $F(T, \tau)$ be the forest induced by all edges of $T$ with weight at most $\tau$ and the connected components of $F(T, \tau)$ induce a clustering; finally, we pick a clustering that can
maximize some quality measure.

The complexity of the algorithm depends on the eigenvector computation together with the Minimum Spanning Tree computation. Classical algorithm such as Prim’s algorithm needs $O(m\log n)$ where $m$ is the number of edges and $n$ is the number of vertices.

**Input:** a constructed graph $G = (V, E)$ with the adjacency matrix $W \in \mathbb{R}^{n \times n}$, degree matrix $D$, embedding dimension $d$, some measure $\textit{quality}$

- $(1, \lambda_1 \ldots, \lambda_d)$: $d + 1$ largest eigenvalues of $M(G) = D(G)^{-1}W(G)$
- $d' = \max\{i: 1 \leq i \leq d, \lambda_i > 0\}$
- $x^{(1)}, \ldots, x^{(d')}$: eigenvectors of $M(G)$ associated with $\lambda_1 \ldots, \lambda_d'$
- For all $e = (u, v) \in E$ do $w(e) = \sum_{i=1}^{d'} |x_u^{(i)} - x_v^{(i)}|$
- $T$: Minimum Spanning Tree of $G$ with respect to $w$
- $C: C(\tau)$ for which $\textit{quality}(C(\tau))$ is maximum over all $\tau \in \{w(e): e \in T\}$

Table 5. Geometric MST Clustering (Brandes et al., 2003)

### 2.5.2 Markov Clustering Algorithm (MCL)

Another algorithm family for graph-based clustering is based on random walks on the graph which can be explained as: random walk that visits a dense cluster will likely not leave the cluster until many of its vertices have been visited. The most successful algorithm in this family is Markov Clustering algorithm (MCL) which was proposed by Dongen (2000).

Similar to the Laplacian matrix in spectral clustering algorithm, MCL defines its own matrix which is called Markov matrix. The Markov matrix $T_G$ is obtained by normalizing the $q$th column of weight matrix $W$ (which is $M_G$) and multiplying the inverse of degree matrix whose diagonals are the sum of column weights of $W$ (which is $D_G^{-1}$), i.e., $T_G = M_G D_G^{-1}$. MCL introduces two operators called expansion and inflation in which expansion coincides with taking the power of a stochastic matrix, while inflation coincides with taking the Hadamard power of a matrix, followed by a scaling step.

Expansion operator corresponds to computing random walks of higher length, which means random walks with many steps. It associates new probabilities with all pairs of nodes, where one node is the point of departure and the other is the destination. Since higher length paths are more common within clusters than between different clusters, the probabilities associated with node pairs lying in the same cluster will, in general, be relatively large as there are many ways of going from one to the other.

Inflation operator will then have the effect of boosting the probabilities of intra-cluster walks and will demote inter-cluster walks.

The result of iterating expansion and inflation is a partition of the graph. There are no longer any paths between the partitions. By controlling the value of inflation parameter, we can control the granularity of final partitions.

The algorithm is presented in Table 6.
**Input:** a constructed graph $G = (V, E)$ with Markov matrix $\mathcal{M}_G$ and degree matrix $\mathcal{D}_G$, expansion parameter $e$ and inflation parameter $r$

- Compute Markov matrix $\mathcal{T}_G = \mathcal{M}_G \mathcal{D}_G^{-1}$
- while $\mathcal{T}_G$ is not fixpoint do
  - $\mathcal{T}_G = \mathcal{T}_G^e$ //expansion operator
  - forall $u \in V$ do
    - forall $v \in V$ do
      - $\mathcal{T}_{uv} = \mathcal{T}_{uv}^r$
    - forall $v \in V$ do
      - $\mathcal{T}_{uv} = \frac{\mathcal{T}_{uv}}{\sum_{w \in E} \mathcal{T}_{uw}}$
  - //inflation operator

- $H$: graph induced by non-zero entries of $\mathcal{T}_G$
- $\mathcal{C}$: clustering induced by connected components of $H$

Table 6. Markov Clustering algorithm (Dongen, 2000)

The complexity of MCL is $O(nk^2)$ where $n$ is the number of vertices in the graph and $k$ is the number of resources allocated per vertex which can be very low without affecting clustering quality.

The MCL algorithm has the following attractive properties:

- It is simple and mathematically elegant which involves in two operations, expansion and inflation.
- It is adaptable. By tuning the expansion and inflation parameter, clusterings on different scales of granularity can be found.

MCL has been applied in a number of different domains with notable successes, especially in computational chemistry and biology. However, MCL has not attracted great attentions in IE or NLP community and probably will in the near future.

### 2.5.3 Miscellaneous

- One of the earliest graph clustering algorithms may be attributed to Kernighan-Lin algorithm (Kernighan and Lin, 1970). Their goal is to partition a graph into two parts of equal size (bisection of the graph) with a minimal number of cutting edges. The algorithm works by iterative improvement, i.e., it starts from an arbitrary bisection and swaps pairs of nodes in order to improve the cost of the partition.

  The algorithm works in passes. In each pass, each node is first marked as “free” and then it enters an inner iteration, a pair of free nodes is selected and swapped in a greedy way, i.e., the pair with the highest gain. The swapped nodes become locked (i.e., not free) afterwards. One pass ends when there are no more free nodes, which means that, as long as there are free nodes, a move is always done, even if it is a worsening one. This is how the algorithm can escape from local optima. At the end of the pass, the algorithm reverts to the partition with the highest gain observed during the pass. All nodes are unlocked and a new pass starts from this partition. The whole algorithm terminates when a pass cannot find a better partition than its starting partition.

  The complexity of this algorithm is $O(n^3)$ where $n$ is the number of vertices in the graph. The high complexity makes it less competitive for real applications. However, the Kernighan-Lin algorithm has been studied, re-discovered, extended by later researchers.
We refer readers to a good survey (53).

- **Karypis and Kumar (1999)** proposed a multi-level approach for bisection graph clustering (two clusters). The basic idea is to reduce the graph by collapsing vertices and edges (coarsening phase), partition the smaller graph (partitioning phase), and then uncoarsen it to construct a partition for the original graph (uncoarsening phase). Formally, consider a weighted graph $G_0 = (V_0, E_0)$, a multi-level graph bisection algorithm consists of the following three phases as shown in Table 7.

| **Input:** a constructed graph $G_0 = (V_0, E_0)$ |
| **Coarsening Phase** |
| The graph $G_0$ is transformed into a sequence of smaller graphs $G_1, ..., G_m$ such that $|V_0| > |V_1| > \cdots > |V_m|$ |
| **Partitioning Phase** |
| A 2-way partition $P_m$ of the graph $G_m = (V_m, E_m)$ is computed that partitions $V_m$ into two parts, each containing half the vertices of $G_0$ |
| **Uncoarsening Phase** |
| The partition $P_m$ of $G_m$ is projected back to $G_0$ by going through intermediate partitions $P_{m-1}, ..., P_0$. |

*Table 7. Karypis and Kumar algorithm (Karypis and Kumar, 1999)*

Their algorithm is named METIS and can be downloaded by the authors’ website: [http://glaros.dtc.umn.edu/gkhome/metis/metis/overview](http://glaros.dtc.umn.edu/gkhome/metis/metis/overview). Although they did not provide an exact overall running complexity of their algorithm, they claimed that the algorithm is extremely fast, one to two orders of magnitude faster than other widely used partitioning algorithms. Furthermore, the partitions produced by METIS are consistently 10% to 50% better than those produced by spectral partitioning algorithms, according to the experiments on a large number of graphs.

The problem of METIS is that it only produces two clusters, and Dhilon et al. (2005) extended the algorithm so that it can produce $K$ clusters.

- **Aksoy and Haralick (1999)** proposed a clustering algorithm that first searches the dense regions in the graph and then merges the dense regions by some criterion. Therefore, the key point in their algorithm is how to find the dense regions in the graph. We refer readers to the original paper for details. The algorithm is presented in Table 8.

| **Input:** a constructed graph $G = (V, E)$ with the adjacency matrix $W \in \mathbb{R}^{n \times n}$ |
| **Step 1.** Search all dense regions in the graph |
| **Step 2.** Merge dense regions if some criterion is satisfied |

*Table 8. Aksoy and Haralick algorithm (Aksoy and Haralick, 1999)*

Although they did not mention the running complexity of their algorithm, a simple analysis shows that the complexity mainly depends on searching the dense regions on a sparse graph which is $O(nm^2)$ in the worse case while the complexity can be almost ignorable in merging dense regions.

The other issue is that their algorithm works on the unweighted graph; however, it is possible to extend their work to the weighted graph.
Flake et al. (2003) proposed a Cut-Clustering algorithm that is based on minimum cut trees. It introduces an artificial node \( t \) which is called artificial sink. The artificial sink is connected to all nodes of \( G \) via an undirected edge of capacity \( \alpha \). The algorithm is presented in Table 9.

**Input:** a constructed graph \( G = (V, E) \) with the adjacency matrix \( W \in \mathbb{R}^{n \times n} \)

Let \( V' = V \cup t \)

For all nodes \( v \in V \)

Connect \( t \) to \( v \) with edge of weight \( \alpha \)

Let \( G' = (V', E') \) be the expanded graph after connecting \( t \) to \( V \)

Calculate the minimum cut tree \( T' \) of \( G' \)

Remove \( t \) from \( T' \)

Return all connected components as the clusters of \( G \)

<table>
<thead>
<tr>
<th>Table 9. Cut Clustering algorithm (Flake et al., 2003)</th>
</tr>
</thead>
</table>

The complexity of this algorithm is dominated by the minimum cut tree computation which is fast, usually in time proportional to the total number of clusters.

Girvan and Newman (2002) proposed a divisive clustering algorithm that involves iterative removal of edges from the graph using some “betweenness” measure. The higher the “betweenness” of the edge, the more probably it lies between clusters and thus the more likely to be removed first. Therefore the key point is how to compute the betweenness. They proposed three methods: Shortest-path betweenness, current-flow betweenness in resistor networks, and random walk betweenness. The other key point in their algorithm is to recalculate the betweenness for all the remaining edges after the step of edge removal. The iteration stops until a clustering quality measure modularity reaches optimal (for definition of modularity measure, refer to section 2.4.2). The algorithm is presented in Table 10.

**Input:** a constructed graph \( G = (V, E) \) with the adjacency matrix \( W \in \mathbb{R}^{n \times n} \)

Step 1. Calculate betweenness scores for all edges in the graph.

Step 2. Find the edge with the highest score and remove it from the graph. If two or more edges tie for highest score, choose one of them at random and remove that.

Step 3. Recalculate betweenness for all remaining edges.

Step 4. Return all connected components as the clusters of \( G \), calculate the value of modularity measure \( Q(C) \)

Repeat from step 2 until \( Q(C) \) reaches an optimal value

<table>
<thead>
<tr>
<th>Table 10. Girvan and Newman algorithm (Girvan and Newman, 2002)</th>
</tr>
</thead>
</table>

The algorithm runs in worst-case time \( O(nm^2) \) on a graph with \( m \) edges and \( n \) vertices or \( O(n^3) \) on a sparse graph.

Newman (2004) proposed an agglomerative hierarchical clustering algorithm that achieves the optimal modularity score in a greedy style. Starting with each vertex as
clusters, the algorithm repeatedly joins clusters together in pairs, choosing at each step the join that results in the greatest increase (or smallest decrease) in modularity score. The progress of the algorithm can be represented as a “dendrogram” and the number of clusters changes from the largest to the smallest. We can select the best clustering by looking for the optimal value of modularity. The algorithm is presented in Table 11.

| Input: a constructed graph $G = (V, E)$ with the adjacency matrix $W \in \mathbb{R}^{n \times n}$
|---|
| Step 1. Initialize the clustering by taking each vertex as a cluster
| Step 2. Join the pair of clusters which leads to the greatest increase (or smallest decrease in modularity score.
| Step 3. Repeat Step 2 until the clustering contains only one cluster
| Step 4. Find the clustering with the optimal modularity score.


The success of this algorithm is that the running complexity is competitive, $O(n(m + n))$ in the worst case or $O(n^2)$ on a sparse graph, comparing to the divisive clustering algorithm proposed by Girvan and Newman.

2.5.4 Summary

We have discussed a collection of algorithms for graph-based clustering now. A natural question arises: “which algorithm do we choose?” A general answer to this question is that no algorithm is a panacea. First, as we mentioned earlier, a clustering algorithm was usually proposed to optimize some quality measure, therefore, it is not fair to compare an algorithm that favors one measure with the other algorithm that favors some other measure. But definitely, an algorithm is poor if it cannot work well on its own measure. Second, there is not a perfect measure that can capture the full characteristics of cluster structures; therefore no algorithm that favors the measure is perfect. Third, there is no definition for so called “best clustering”. The “best” depends on applications, data characteristics, granularity and so on (e.g., for a news archive, some application prefers to cluster the articles into topics, some other application prefers to cluster them chronologically with varied granularity). All the above reasons can explain why the graph clustering algorithms have been discovered, re-discovered, and extended in various research communities. Another question is that if we have a ground-truth clustering, can we compare the algorithms by evaluating the output clusterings? Yes, but how? We will provide answers in the next section.

2.6 Evaluation

We have discussed various measures (objective functions) in section 2.4 to obtain or approximately obtain the “optimal” quality of a clustering. However, high values in those measures do not necessarily translate into effectiveness in real applications if gold standard (ground truth) clustering is available. Obviously we need different set of measures to evaluate the quality of clustering relative to the gold standard. To distinguish the two types of measures, we refer to the measures in section 2.4 as internal (intrinsic) and the measures discussed in
this section as external (extrinsic). The external measures tend to be more reliable but much more expensive because the gold standard is usually set up by expensive human labors including human annotators, human assessors, and human adjudicators.

In this section, we discuss the evaluation problem by answering the following two questions: (1) Are there any formal constraints (properties, criteria) that an ideal extrinsic measure should satisfy? (2) Do the extrinsic measures proposed so far satisfy the constraints?

In the discussion followed, we refer to the system clusters as CLUSTERS, and the reference clusters as CLASSES.

2.6.1 Formal Constraints on Evaluation Measures

There are some intuitions about what makes a better clustering, for example, we would prefer a cluster containing all “clean” items rather than a cluster containing most of “clean” items with a few “noise” items; for the other example, we prefer the “clean” items showing up in a single cluster rather than dispersing among the clusters. These two intuitions correspond to the two essential criteria (homogeneity and completeness respectively) that were proposed by Rosenberg and Hirschberg (2007). As they pointed out, homogeneity and completeness run roughly in opposition, i.e., increasing the homogeneity often results in decreasing the completeness, and in two extreme cases, a clustering with all singletons obtains perfect homogeneity but worst completeness and a clustering with a single cluster obtains perfect completeness but worst homogeneity.

Dom (2001) developed a parametric technique for describing the quality of a clustering and proposed five “desirable properties” based on the parameters. As a step further, Rosenberg and Hirschberg (2007) extended the parameter set and proposed another two desirable properties. The parameters in Dom (2001) include: the number of classes, the number of “noise” (containing items equally from each class) and “useful” (otherwise) clusters, and two components of error mass $E_1$ (evenly distributed across each pair of non-matching useful class/cluster pairs), and $E_2$ (distributed across every noise cluster/useful class pair). The extended parameters introduced in Rosenberg and Hirschberg (2007) include: the number of “noise” (containing items equally from each cluster) and “useful” (otherwise) classes, $E_3$ (distributed across every useful cluster/noise class pair). For the details of the “desirable properties”, we refer readers to the original paper of Dom (2001) and Rosenberg and Hirschberg (2007) respectively, but basically they capture the idea that a clustering is worse whenever

- The number of useful clusters varies away from the number of classes
- The number of noise clusters increases
- The error mass $E_1, E_2, E_3$ increases

Meila (2003) listed 12 properties associated with the measure he proposed. However, only a few of them are directly related to the quality aspect captured by a measure, specifically, Property 2 states that the quality of the measure only depends on the relative sizes of clusters rather than the number of data points (thus this property is called $n$-invariant); Property 5 states that splitting or merging smaller clusters has less impact than splitting or merging larger ones; Property 12 states that the impact of splitting or merging clusters is limited to only those clusters involved (thus this property is called locality).
Recently, Amigo et al. (2008) proposed four formal constraints which, as they claimed, have the following merits: (1) they are intuitive and can clarify the limitations of each measure (2) it is possible to prove formally which measures satisfy which properties (3) the constraints can discriminate measure families, indicating the limitations of each measure family rather than individual measure variants.

The four formal constraints in Amigo et al. (2008) include two common constraints (homogeneity and completeness) as proposed in Rosenberg and Hirschberg (2007), together with another two new constraints:

- **Rag bag**: the intuition is that introducing disorder into a disordered cluster is less harmful than introducing disorder into a clean cluster. In practice, it is helpful to have a “rag bag” of items that cannot be grouped into any known classes (so it can be named as miscellaneous, other, or unclassified). Thus, we prefer a clean cluster with a “rag bag” cluster rather than a cluster with a dominant class plus additional noise.

- **Cluster size vs. quantity**: the intuition is that a small error in a big cluster is more preferable than a large number of small errors in small clusters. Thus we prefer a large cluster with one item left out rather than many small clusters with isolated items. This property is partially related with Property 2 in Meila (2003).

These four formal constraints can be illustrated quite intuitively by Figure 5 (a,b,c,d respectively).

Figure 5. Illustrations of four formal constraints (Amigo et al., 2008)

Amigo et al. (2008) also compared their four constraints with the constraints proposed in Dom (2001) and in Meila (2003), and reached the conclusion that theirs have advantages over the others, specifically,

- The four constraints can describe all the important properties in Dom (2001) and Meila (2003), but neither Dom nor Meila’s properties can describe the “rag bag” constraint of Amigo et al. (2008).
- It is not easy to prove formally that a measure satisfies Dom’s constraints; by contrast, Amigo’s constraints can be formally verified for each measure.

2.6.2 Evaluation Measures

Now we review various evaluation measures, discuss the advantages and disadvantages, and investigate whether they satisfy the formal constraints.
For the convenience of discussion, we assume that the data set consists of $N$ data points, the system clustering is $\mathcal{C} = (C_1, \ldots, C_K)$, and the gold standard (reference clustering) is $\mathcal{R} = (R_1, \ldots, R_M)$.

We also categorize the evaluation measures into families, as shown in Amigo et al. (2008).

### 2.6.2.1 Measures Based on Set Mapping

1. **Purity (Zhao and Karypis, 2001)**

   To compute purity, each cluster is assigned to the class which is the most frequent in the cluster, and then sum the number of matching items for each cluster, dividing by $N$, i.e.,

   $$
purity = \frac{1}{N} \sum_{i=1}^{K} \max_{1 \leq j \leq M} |C_i \cap R_j|$$

   (42)

   The larger the value of purity, the better the clustering is. A bad clustering has a purity value near to 0, and a perfect clustering has a purity of 1. Unfortunately, this measure is easy to “cheat” because high purity is likely to be obtained by enlarging the number of clusters; in particular, purity is 1 if each cluster is a singleton.

2. **Inverse purity**

   The formula of inverse purity is similar to purity and is defined as follows:

   $$inverse\_purity = \frac{1}{N} \sum_{i=1}^{M} \max_{1 \leq j \leq K} |R_i \cap C_j|$$

   (43)

   The larger the value of inverse purity, the better the clustering is. Unfortunately, this measure is also easy to “cheat” because high inverse purity is likely to be obtained by reducing the number of clusters; in particular, inverse purity is 1 if the clustering contains only one cluster with all items.

3. **F-measure**

   F-measure combines the concepts of the precision and recall from information retrieval (Larsen and Aone, 1999).

   The precision, recall, F-measure for class $R_i$ and cluster $C_j$ are defined as follows:

   $$precision(R_i, C_j) = \frac{|R_i \cap C_j|}{|C_j|}$$

   (44)

   $$recall(R_i, C_j) = \frac{|R_i \cap C_j|}{|R_i|}$$

   (45)

   $$F\_Measure(R_i, C_j) = \frac{2 \times precision(R_i, C_j) \times recall(R_i, C_j)}{precision(R_i, C_j) + recall(R_i, C_j)}$$

   (46)

   Intuitively, $F\_Measure(R_i, C_j)$ measures how good a class $R_i$ can be described by
a cluster \( C_i \) and the success of capturing a class \( R_i \) is measured by using the “best”
cluster \( C_j \) for \( R_i \), i.e., the \( C_j \) that maximizes \( F\_Measure(R_i, C_j) \).

The F-measure for the entire clustering is then defined as

\[
F\_Measure = \frac{1}{N} \sum_{i=1}^{M} |R_i| \max_{i \leq j \leq K} F\_Measure(R_i, C_j) \tag{47}
\]

F-measure has a significant advantage over purity or inverse purity since it considers
both of them, however, it suffers seriously from the “problem of matching” as indicated in
Rosenberg and Hirschberg (2007), i.e., it only considers the contributions from those clusters
that are matched to a target class. A counter-example is given in Figure 6 (A and B represent
two clusterings, the shapes represent classes). The two clusterings produce exactly the same
F-measure which is 0.6, but obviously B is better than A in terms of both homogeneity (each
cluster contains fewer classes) and completeness (each class is contained in fewer clusters).

![Figure 6. A counter-example that shows the drawback of F-measure (Rosenberg and
Hirschberg, 2007)](image)

### 2.6.2.2 Measures Based on Pair Counting

Another important measure family is based on counting the pairs of points on which two
clusterings agree or disagree. Any pair of data points from the total of \( \frac{N(N-1)}{2} \) distinct pairs
falls into one of the following four groups:

- \( S_{11} \): the set of pairs of items that are in the same cluster and class,
- \( S_{12} \): the set of pairs of items that are in the same cluster and different class,
- \( S_{21} \): the set of pairs of items that are in different cluster and the same class,
- \( S_{22} \): the set of pairs of items that are in different cluster and class.

We use \( a, b, c, d \) to represent the size of the four sets respectively.

1. Rand index (Rand, 1971)

\[
\text{Rand index} = \frac{a + d}{a + b + c + d} \tag{48}
\]

In the formula, \( a \) and \( d \) can be interpreted as agreement, \( b \) and \( c \) as disagreements.
The Rand index lies between 0 and 1. If the clustering is perfect, the Rand index is 1.

A problem with the Rand index is that the expected value of the Rand index of two
random clusterings does not take a constant value.

2. Adjusted rand index (Hubert and Arabie, 1985)
Adjusted rand index = \frac{2 \times (a + d - b + c)}{(a + b)(d + b) + (a + c)(d + c)} \tag{49}

It takes the generalized hypergeometric distribution as the model of randomness, i.e., the two clusterings are picked at random such that the number of objects in the class and clusters are fixed.

(3) Jaccard Coefficient \hspace{0.5cm} \text{(Milligan et al., 1983)}

\begin{equation}
\text{jaccard Coefficient} = \frac{a}{a + b + c} \tag{50}
\end{equation}

(4) Folks and Mallows FM\hspace{0.5cm} \text{(Fowlkes and Mallows, 1983)}

\begin{equation}
\text{Folks and Mallows FM} = \sqrt{\frac{a}{a + b} \times \frac{a}{a + c}} \tag{51}
\end{equation}

2.6.2.3 Measures Based on Entropy

Let \( A \) be the contingency matrix such that \( A = \{a_{ij}\} \) where \( a_{ij} \) is the number of data points that are in class \( i \) and are assigned to cluster \( j \).

Then we have:

\( P(i,j) = \frac{a_{ij}}{N} \): the probability of an item in class \( i \) and is assigned to cluster \( j \),

\( P(i) = \frac{|R_i|}{N} \): the probability of an item in class \( i \),

\( P(j) = \frac{|C_j|}{N} \): the probability of an item in cluster \( j \)

(1) Entropy \hspace{0.5cm} \text{(Steinbach et al., 2000)}

The entropy of a cluster \( C_j \) is defined as

\begin{equation}
\text{entropy}(C_j) = -\sum_{i=1}^{M} P(i,j) \times \log P(i,j) \tag{52}
\end{equation}

The total entropy is then computed by averaging the entropy of all clusters:

\begin{equation}
\text{entropy}(C) = \frac{1}{N} \sum_{j=1}^{K} |G_j| \times \text{entropy}(C_j) \tag{53}
\end{equation}

The lower the value of entropy, the better the clustering is. The entropy measure also has the same drawback as purity, and it obtains lowest value 0 when the clustering contains all singletons.

(2) Mutual information\hspace{0.5cm} \text{(Xu et al., 2003)}

\begin{equation}
MI(C, R) = \sum_{i=1}^{M} \sum_{j=1}^{K} P(i,j) \times \log \frac{P(i,j)}{P(i)P(j)} \tag{54}
\end{equation}
A normalized version of the mutual information is defined in (Strehl and Ghosh, 2002)

\[ NMI(C, \mathcal{R}) = \frac{MI(C, \mathcal{R})}{\sqrt{H(C)H(\mathcal{R})}} \] (55)

The higher the value of mutual information which means the system clustering shares more information with the reference clustering, the better the clustering is.

(3) Variation of information (VI) (Meila, 2003)

The other way to compute the entropy of the system clustering is defined as

\[ entrop(C) = -\sum_{j=1}^{K} P(j)\log(P(j)) \] (56)

Similarly, the entropy of the reference clustering is

\[ entrop(\mathcal{R}) = -\sum_{i=1}^{M} P(i)\log(P(i)) \] (57)

The conditional entropy of system clustering given reference clustering is

\[ entrop(C | \mathcal{R}) = -\sum_{i=1}^{M} \sum_{j=1}^{K} \frac{a_{ij}}{N} \log \frac{a_{ij}}{|\mathcal{R}|} \] (58)

The conditional entropy of reference clustering given system clustering is

\[ entrop(\mathcal{R} | C) = -\sum_{j=1}^{K} \sum_{i=1}^{M} \frac{a_{ij}}{N} \log \frac{a_{ij}}{|C|} \] (59)

Then VI measure is defined as

\[ VI(C, \mathcal{R}) = entrop(\mathcal{R} | C) + entrop(C | \mathcal{R}) \] (60)

The VI measure captures the characteristics of homogeneity and completeness simultaneously. When the clustering is the least homogeneous, the class distribution within each cluster is equal to the overall class distribution, and \( entrop(\mathcal{R} | C) \) is maximized, i.e., \( entrop(\mathcal{R} | C) = entrop(\mathcal{R}) \). When the clustering is the least complete, \( entrop(C | \mathcal{R}) \) is maximized, i.e., \( entrop(C | \mathcal{R}) = entrop(C) \). When the clustering becomes more homogeneous and complete, \( entrop(\mathcal{R} | C) \) and \( entrop(C | \mathcal{R}) \) decrease to 0. The perfect clustering achieves 0 for VI measure, thus, the lower the value of VI, the better the clustering is.

Meila (2003) has proved 12 properties that VI measure satisfies. However, the VI measure is hard to compare among datasets since it depends on \( N \) which is the number of data points. To solve this problem, Reichart and Rappoport (2009) proposed a normalized version VI, called NVI measure.

(4) V-measure (V) (Rosenberg and Hirschberg, 2007)

The V measure uses homogeneity (\( h \)) and completeness (\( c \)) terms as follows:

\[ h = \begin{cases} 1 & \text{entropy}(\mathcal{R}) = 0 \\ 1 - \frac{\text{entropy}(\mathcal{R} | C)}{\text{entropy}(\mathcal{R})} & \text{entropy}(\mathcal{R}) \neq 0 \end{cases} \] (61)
Both $h$ and $c$ lie in $[0,1]$. For the most homogeneous and most complete clustering, $\text{entropy}(R|C) = 0$ and $\text{entropy}(C|R) = 0$ respectively, and $h$ and $c$ reaches maximum. Since $V$ is the harmonic mean of $h$ and $c$, $v$ also lies in $[0,1]$, thus it is independent of the size of the dataset and can be used to compare the performance of clustering algorithms across datasets. The higher the $V$ is, the better the clustering is.

2.6.2.4 Measures Based on Editing Distance

(1) Editing distance (Pantel and Lin, 2002)

The editing distance is defined as the number of operations required to transform a system clustering into reference clustering. Pantel and Lin (2002) define the following three operations: (a) merge two clusters; (b) move an element from one cluster to another; (c) copy an element from one cluster to another. The lower the value of editing distance which means fewer operations for the transformation, the better the clustering is.

2.6.2.5 Measures for Coreference Resolution

The following three measures MUC F-measure, B-Cubed F-measure, ECM F-measure have been well known in the information extraction field, specifically, the coreference resolution applications.

(1) MUC F-measure (Vilain et al., 1995)

- Let $P(R_i)$ be a partition of $R_i$ relative to the system clustering $C = (C_1, \ldots, C_K)$ where $R_i$ is a reference cluster. The partition is constructed by intersecting $R_i$ and those clusters in $C$. For example, if $R_i = \{A, B, C, D\}$ and $C = \{\{A, B\}, \{C\}, \{D\}, \{E\}\}$, then $P(R_i) = \{\{A, B\}, \{C\}, \{D\}\}$.

- Let $c(R_i)$ be the minimal number of “correct” links (node-node connection) necessary to generate $R_i$. $c(R_i) = |R_i| - 1$. For example, $R_i = \{A, B, C, D\}$, at least three links are needed, $\{A \rightarrow B\}$, $\{B \rightarrow C\}$, $\{C \rightarrow D\}$.

- Let $m(R_i)$ be the number of “missing” links in the system clustering relative to $R_i$. $m(R_i) = |P(R_i)| - 1$. Continue the above example. There are two $(|P(R_i)| - 1 = 2)$ “missing” links that groups the items into $R_i$, $\{B \rightarrow C\}$ and $\{C \rightarrow D\}$.

- The recall of $R_i$ is defined as

$$\text{recall}(R_i) = \frac{c(R_i) - m(R_i)}{c(R_i)} = \frac{|R_i| - 1 - (|P(R_i)| - 1)}{|R_i| - 1} = \frac{|R_i| - |P(R_i)|}{|R_i| - 1}$$

(64)

The recall of the reference clustering is then defined by extending the recall of $R_i$ to the entire set of $R$. 

27
\[
recall(C) = \frac{\sum_{i=1}^{M}(|R_i| - |P(R_i)|)}{\sum_{i=1}^{M}(|R_i| - 1)}
\] (65)

- The precision is computed by switching the role of reference and system of the above notations, namely, \(P(C_i)\) is a partition of \(C_i\) relative to the reference clustering \(R\),

\[
precision(C_i) = \frac{c(C_i) - m(C_i)}{c(C_i)} = \frac{|C_i| - (|P(C_i)| - 1)}{|C_i| - 1} = \frac{|C_i| - |P(C_i)|}{|C_i| - 1}
\] (66)

\[
precision(C) = \frac{\sum_{i=1}^{K}(|C_i| - |P(C_i)|)}{\sum_{i=1}^{K}(|C_i| - 1)}
\] (67)

- The F-measure is then defined

\[
F(C) = \frac{2 \cdot precision(C) \cdot recall(C)}{precision(C) + recall(C)}
\] (68)

The drawbacks of MUC F-measure include (Bagga and Baldwin, 1998)

- It does not give any credits for separating out singleton clusters, as we can see that if \(R_i\) or \(C_i\) contains only one item, the denominator and numerator of \(recall(R_i)\) or \(precision(C_i)\) simply obtain 0, and thus they do not contribute to the final score of recall or precision.
- All errors are considered to be equal which may not be desirable sometimes. For example, if

\[
R = \{(1,2,3,4,5), \{6,7\}, \{8,9,A,B,C\}\}
\]

\[
C_1 = \{(1,2,3,4,5,6,7), \{8,9,A,B,C\}\}
\]

\[
C_2 = \{(1,2,3,4,5,8,9,A,B,C), \{6,7\}\}
\]

Using MUC F-measure, the two system clusterings obtain exactly the same score \(recall(C_1) = recall(C_2) = 1\), \(precision(C_1) = precision(C_2) = 0.9\), \(F(C_1) = F(C_2) = 0.947\) but obviously the second clustering is worse since it makes more wrong items group together.

(2) B-Cubed F-measure (Bagga and Baldwin, 1998)

Unlike any other measures, B-Cubed measure evaluates the clustering by summing the score of each item in the clustering.

For each item \(i\), the precision and recall are

\[
precision_i = \frac{\# \text{correct items in the system cluster containing item } i}{\# \text{items in the system cluster containing item } i}
\] (69)

\[
recall_i = \frac{\# \text{correct items in the system cluster containing item } i}{\# \text{items in the reference cluster containing item } i}
\] (70)

\[
precision(C) = \sum_{i=1}^{N} w_i precision_i
\] (71)

\[
recall(C) = \sum_{i=1}^{N} w_i recall_i
\] (72)
where $w_i$ is the weight assigned to item $i$ (e.g., $w_i = 1/N$).

Obviously, B-Cubed measure overcomes the two drawbacks of MUC measure. Particularly, in the example which illustrates the second drawback of MUC, B-Cubed obtain $\text{recall}(C_1) = \text{recall}(C_2) = 1, \text{precision}(C_1) = 0.762, \text{precision}(C_2) = 0.583$, thus, $C_2$ is worse than $C_1$ which is exactly what we desire.

However, as pointed out by Luo (2005), B-Cubed may give multiple credits to a single item and it shows counter-intuitive results in two extreme cases: if the system clustering contains only one cluster with all the items, B-Cubed recall is 100%; if the system clustering contains all singleton clusters, B-Cubed precision is 100%.

(3) ECM F-measure (Luo, 2005)

It seeks an optimal alignment between the system clustering and the reference clustering such that the similarity of them is maximized. The problem is thus modeled as a classical maximum bipartite matching problem: each cluster in $C$ and $R$ is a vertex, and the vertex pair $(C_i, R_i)$ is connected by an edge with the similarity weight $\phi(C_i, R_i)$. The similarity weight can be obtained by finding the common items in $C_i$ and $R_i$, i.e., $\phi(C_i, R_i) = |C_i \cap R_i|$, or alternatively, $\phi(C_i, R_i) = \frac{2|C_i \cap R_i|}{|C_i| + |R_i|}$ The problem can be solved by an efficient algorithm named Kuhn-Munkres algorithm (Kuhn, 1955; Munkres, 1957).

### 2.6.2.6 Satisfaction of Formal Constraints in Various Measures

In this section, we follow the four formal constraints proposed by Amigo et al. (2008) to validate the quality of all the above mentioned measures and we also extend the discussions to more measures that were not covered in Amigo et al. (2008), namely, adjusted rand index, V measure, MUC F-measure and ECM F-measure. The results are shown in Table 12.

<table>
<thead>
<tr>
<th>Formal constraints</th>
<th>Homogeneity</th>
<th>Completeness</th>
<th>Rag bag</th>
<th>Cluster size vs. quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1" alt="Homogeneity" /></td>
<td><img src="image2" alt="Completeness" /></td>
<td><img src="image3" alt="Rag bag" /></td>
<td><img src="image4" alt="Cluster size vs. quantity" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measures based on set matching</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Purity</td>
<td>0.71 0.78 ✓</td>
<td>0.78 0.78 ×</td>
<td>0.55 0.55 ×</td>
</tr>
<tr>
<td>Inverse purity</td>
<td>0.78 0.78 ×</td>
<td>0.78 0.78 ×</td>
<td>1 1 ×</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.63 0.63 ×</td>
<td>0.62 0.62 ×</td>
<td>0.61 0.61 ×</td>
</tr>
<tr>
<td>OK</td>
<td>FAIL</td>
<td>FAIL</td>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measures based on pair counting</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rand index</td>
<td>0.68 0.7 ✓</td>
<td>0.68 0.7 ✓</td>
<td>0.72 0.72 ×</td>
</tr>
<tr>
<td>Adjusted rand*</td>
<td>0.25 0.28 ✓</td>
<td>0.24 0.31 ✓</td>
<td>0.4 0.4 ×</td>
</tr>
<tr>
<td>Jaccard</td>
<td>0.31 0.32 ✓</td>
<td>0.31 0.35 ✓</td>
<td>0.37 0.37 ×</td>
</tr>
<tr>
<td>F&amp;M</td>
<td>0.47 0.49 ✓</td>
<td>0.47 0.52 ✓</td>
<td>0.61 0.61 ×</td>
</tr>
</tbody>
</table>
Table 12 shows that none of the measures except B-Cubed F-measure can satisfy all the four constraints. The measure family based on set matching can satisfy homogeneity and (cluster size vs. quantity), but not completeness and rag bag. The measure family based on pair counting (including adjusted rand index) can satisfy homogeneity and completeness, but not rag bag and (cluster size vs. quantity). The two measures (VI and V measure) in the entropy measure family seem to work better than others, but still fail the rag bag constraint. It is worth noting that MUC F-measure is not that bad as far as the constraint satisfaction is concerned (as it only fails rag bag constraint) although as we analyzed earlier, it does suffer some drawbacks. We also find that the ECM F-measure fails three constraints: homogeneity, completeness and rag bag. Further analysis shows that the optimal matching of ECM F-measure will likely to ignore the “exceptional” clusters (the middle small clusters in the examples of illustrating homogeneity and completeness, and the top mixed cluster in the example of illustrating rag bag) which speak the problem. Therefore, ECM F-measure also has drawbacks even though it has been claimed to overcome the drawbacks of B-Cubed F-measure.

2.7 Open Problems and Future Directions

To summarize section 2, although graph-based clustering has been studied for decades, the theoretical foundations of graph clustering are not fully explored. Besides some open problems discussed earlier (e.g., parameter selection in graph construction, properties that a good clustering should exhibit), we list some more open problems as follows:

1 Measures with * are our extensions. ✓ means satisfied, and ✗ means not satisfied. OK means at least one of the measures in the family satisfies the constraint, FAIL means none of the measures in the family satisfy the constraint.
Scalability: It becomes a critical issue because for one hand, the graphs in real applications (e.g., information network, social network, web graph) are growing rapidly; for the other hand, the graphs are also changing dynamically (e.g., web pages are added, modified, or removed daily on the web). In the former case, scalability means that the computational resources (e.g., running time and consumed memories) only grow moderately when the graph grows rapidly; in the latter case, it means that the clustering algorithms can dynamically adjust with the changing environment. Promising directions include the development of parallel and distributed graph-based clustering algorithms, incremental graph clustering algorithms.

Stability: By stability, we mean that the graph clustering algorithms should produce stable clustering if the graph undergoes perturbations, e.g., insertion or removal of a few edges and/or vertices. Algorithms that produce stable clustering are preferred and therefore stability-based methods are useful tools for algorithm (model) selection in clustering if we do not have ground-truth clustering for algorithm (model) assessment. However, up to now, we still lack a theoretical understanding for stability methods, in particular, in is unclear in which situations stability works and what the mechanism is which makes it a successful tool in those situations.

Statistical significance: The problem is that even for a random graph, a graph-based clustering algorithm can always find a clustering that looks like real (vertices in a subgraph are densely connected and sparsely connected to the rest of the graph). Reichardt and Bornholdt (2006) have shown that purely random graphs can display intrinsic modularity and may be partitioned yielding high values of modularity. They stress that statistically significant modularity must exceed the expectation values of modularity from a suitable null model of the graph. Although they have made progress by providing theoretical results for modularity, the statistical significance problem still needs to be explored from the views other than modularity.

3. Coreference Resolution: an Application in Information Extraction

We select coreference resolution as our case study of applying the graph based clustering methodology. Typically, coreference resolution is the problem of identifying which noun phrases (NPs, or mentions) refer to the same real-world entity in text. An entity is an object or a set of objects in the real world such as person, organization, facility, while a mention is a textual reference to an entity. In the following example as shown in Table 13, mentions are underlined.

The American Medical Association voted yesterday to install the heir apparent as its president-elect, rejecting a strong, upstart challenge by a District doctor who argued that the nation’s largest physicians’ group needs stronger ethics and new leadership.

Table 13. An example of entity coreference resolution (Luo et al., 2004)

In the underlined mentions, “its” and “group” refer to their antecedent “American Medical Association”, thus they should be grouped as an entity.

Recently, coreference resolution has been redefined in a different problem which is
called event coreference resolution. To differentiate it from the traditional coreference resolution, we call the traditional coreference resolution as entity coreference resolution. An event is a specific occurrence involving participants. An event mention is a textual reference to an event which includes a distinguished trigger (the word that most clearly expresses an event occurs) and involving arguments (entities/temporal expressions that play certain roles in the event). In the following example as shown in Table 14, for each event mention (EM), triggers are surrounded by curly brackets and arguments are underlined.

| Example | EM1: Rudolph Giuliani will {wed} his companion, Judith Nathan, on May 24 in the ex-mayor’s old home. | EM2: Mayor Michael Bloomberg, will perform the {ceremony}. | EM3: The Giuliani-Nathan {nuptials} will be a first for Bloomberg, who is making an exception from his policy of not performing weddings. |

Table 14. An example of event coreference resolution (Chen et al., 2009)

In the above example, EM1, EM2 and EM3 corefer with each other because they have the same event type and subtype (LIFE:MARRY) indicated by a verb trigger “wed” and two noun triggers “ceremony” and “nuptials” respectively. Furthermore, the two persons “Rudolph Giuliani” and “Judith Nathan” involving in the “MARRY” event in EM1 corefer with “Giuliani” and “Nathan” in EM3 respectively.

3.1 A Parallel Comparison between Entity Coreference Resolution and Event Coreference Resolution

In this section, we show the similarities and differences between entity coreference resolution and event coreference resolution. They are similar because (1) the problem descriptions are similar; (2) the mathematical interpretations are similar; (3) the procedures to solve the two problems are similar; (4) they can be solved in graph-based clustering framework. They are different because (1) entity and event have different attributes and values. We carry out the discussions based on the above arguments.

Formally, entity coreference resolution can be formulated as a clustering problem, i.e., grouping all the mentions of entities into equivalent clusters so that all the mentions in a given cluster refer to an entity. Event coreference resolution can be quite similarly defined, i.e., grouping all the mentions of events into equivalent clusters so that all the mentions in a given cluster refer to an event.

We can use two similar sets of notation to mathematically interpret the two clustering problems and once we define the version of notation for entity coreference resolution (as shown in Table 15), we can obtain the notation for event coreference resolution simply by replacing the key word “entity” with “event”.

Let $I$ be the set of positive integers. Let $A$ be a set of attributes and $V$ be a set of values. Some attributes may have no values and some attributes may have one or more values. Any information about an entity is a subset of $A \times V$, and the same applies to an entity mention.
Let $M$ be the set of possible entity mentions in a document $D$. Let $<em_i \in M \mid i = 1, \ldots, N> \in M$ be the $N$ entity mentions in the document $D$ listed in the order in which they occur in the document.

Let $E$ be the set of possible entities in the document $D$. Let $<e_j \in E \mid j = 1, \ldots, K> \in E$ be the $K$ entities.

The goal of entity coreference resolution is to construct a function $f: I \rightarrow I$, mapping entity mention index $i \in I$ to entity index $j \in I$.

### Table 15. Notation for entity coreference resolution

The above notation in fact defines within-document entity coreference resolution, and we can extend it to cross-document entity coreference resolution by replacing “document $D$” with “corpus $C$”. However, a problem for cross-document coreference resolution is that it is not easy to determine the order the entity mentions occur in the corpus.

The above notation does not tell anything about what attributes and values should be included in the attribute set $A$ and value set $V$. Therefore, we can add an infinite number of attributes and values into the sets as long as they can characterize an entity (entity mention) or event (event mention) from some aspects. As an example, we discuss the Entity Detection and Recognition (EDR) task and Event Detection and Recognition (VDR) task specified in the Automatic Content Extraction (ACE) 2005 program.

An entity of EDR has the following attributes and values as shown in Table 16 and we refer readers to (NIST, 2005) for details.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>FAC(Facility), GPE(Geo-Political), LOC(Location), ORG(Organization), PER(Person), VEH(Vehicle), WEA(Weapon)</td>
</tr>
<tr>
<td>subtype</td>
<td>values omitted due to the large quantities</td>
</tr>
<tr>
<td>class</td>
<td>SPC (A particular, specific and unique real world entity)</td>
</tr>
<tr>
<td></td>
<td>GEN(A kind or type of entity rather than a specific entity)</td>
</tr>
<tr>
<td></td>
<td>NEG(A negatively quantified, usually generic entity)</td>
</tr>
<tr>
<td></td>
<td>USP(An underspecified entity, e.g., modal/uncertain/…)</td>
</tr>
<tr>
<td>ems (a set of entity mentions)</td>
<td>can be enumerated given corpora</td>
</tr>
</tbody>
</table>

### Table 16. Attributes and values of an entity (NIST, 2005)

An entity mention has the following attributes and values as shown in Table 17.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>NAM:A proper name reference to the entity</td>
</tr>
<tr>
<td></td>
<td>NOM: A common noun reference to the entity</td>
</tr>
<tr>
<td></td>
<td>PRO: A pronominal reference to the entity</td>
</tr>
<tr>
<td>head</td>
<td>can be enumerated given corpora</td>
</tr>
</tbody>
</table>

### Table 17. Attributes and values of an entity mention (NIST, 2005)

Besides the above well-defined attributes in (NIST, 2005), we can further extend the set of attributes, for example, an entity mention can have attributes of gender (with possible values of female, male, neutral, and unknown), number (with possible values of singular,
An event of VDR has the following attributes and values as shown in Table 18.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Life, Movement, Transaction, Business, Conflict, Contact, Personnel, Justice</td>
</tr>
<tr>
<td>subtype</td>
<td>values omitted due to the large quantities, refer to (NIST, 2005)</td>
</tr>
<tr>
<td>arguments</td>
<td>can be enumerated given corpora (they are entities, temporal expressions and values)</td>
</tr>
<tr>
<td>ems (a set of event mentions)</td>
<td>can be enumerated given corpora</td>
</tr>
<tr>
<td>polarity</td>
<td>NEGATIVE and POSITIVE</td>
</tr>
<tr>
<td></td>
<td>An event is NEGATIVE if it is explicitly indicated that the event did not occur, otherwise, the event is POSITIVE.</td>
</tr>
<tr>
<td>modality</td>
<td>ASSERTED and OTHER</td>
</tr>
<tr>
<td></td>
<td>An event is ASSERTED if it is mentioned as if it were a real occurrence, otherwise it is OTHER.</td>
</tr>
<tr>
<td>genericity</td>
<td>SPECIFIC and GENERIC</td>
</tr>
<tr>
<td></td>
<td>An event is SPECIFIC if it is a single occurrence at a particular place and time, or a finite set of such occurrences; otherwise, it is GENERIC.</td>
</tr>
<tr>
<td>tense</td>
<td>PAST, FUTURE, PRESENT and UNSPECIFIED</td>
</tr>
<tr>
<td></td>
<td>The PAST events occurred prior to the anchor time; the FUTURE events have not yet occurred at the anchor time; the PRESENT events occur at the anchor time; all the other events are UNSPECIFIED.</td>
</tr>
</tbody>
</table>

Table 18. Attributes and values of an event (NIST, 2005)

An event mention of VDR has the following attributes and values as shown in Table 19.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>trigger</td>
<td>can be enumerated given corpora</td>
</tr>
<tr>
<td>arguments</td>
<td>can be enumerated given corpora (they are mentions of entities, temporal expressions or values)</td>
</tr>
</tbody>
</table>

Table 19. Attributes and values of an event (NIST, 2005)

The above attributes play important roles in solving entity coreference resolution and event coreference resolution. Basically, they can be incorporated into learning models as features which compute how likely pairs of mentions corefer.

For both entity coreference resolution and event coreference resolution, we can apply a two step procedure to solve the problem: (1) a classification step that computes how likely one mention corefers with the other and (2) a clustering step that groups the mentions into

---

2 The “value” here has a specific meaning, ACE defines several types of values, e.g., contact information like e-mail, phone-number, url; numeric like money and percent
clusters such that all mentions in a cluster refer to the same entity or event.

Both problems can be modeled as graphs, in which the nodes represent all the entity/event mentions in a document and the edge weights indicate the coreference likelihood between two entity/event mentions. The graph notation for entity coreference resolution is defined in Table 20 and a similar version can be defined for event entity coreference by replacing the keyword “entity” with “event”. We continue to use some of the notation defined in Table 15, namely, $M$ is the set of possible entity mentions.

Let $\text{coref}: M \times M \rightarrow [0,1]$ be the function that computes the coreference likelihood between two entity mentions $em_i, em_j \in M$.

Let $T = \{t_k: 1 \leq k \leq K\}$ be $K$ entity types.

Thus for each entity type $k$, we have a graph $G_k(V_k, E_k)$, where $V_k = \{em_m | em_m, \text{type} = t_k, em_m \in M\}$ and $E_k = \{(em_i, em_j, \text{coref}(em_i, em_j)) | em_i, em_j \in M\}.$

The goal of entity coreference resolution in graph-based framework is to cluster each $G_k(V_k, E_k)$ into subgraphs $G'_k(V'_k, E'_k)$ such that the mentions in the subgraph are densely connected while sparsely connected across subgraphs.

Table 20. Graph notation for entity coreference resolution

It is worth noting that for computing edge weight $\text{coref}$ in the graph, we can apply similar techniques that have been used in the classification step of the two step procedure. The major difference is located in the second step, i.e., clustering.

Last but not least, both problems assume that mentions (either entity mentions or event mentions) have been detected ahead of coreference resolution, however, since mention detection may also introduce great errors, researchers have proposed joint models that intend to solve the two problems in a simultaneous style, e.g., a Learning as Search Optimization (LaSO) framework proposed in (DaumeIII and Marcu, 2005).

To summarize this section, because of the similarities shared between the two problems, the major techniques (algorithms and evaluation measures) can be applied to each other. However, since event has much more complex structure (trigger and arguments) than entity, we have quite different features for the leaning models.

In the following two sections, we will first briefly present a short literature review on entity coreference resolution and event coreference resolution respectively. We will then focus on the graph-based clustering methodology for solving these two problems. Most importantly, we will compare graph-based clustering algorithm with other proposed algorithms such that we can know why graph-based clustering algorithm has achieved state-of-the-art performance.

### 3.2 Entity Coreference Resolution

The research of entity coreference resolution has shifted from earlier knowledge-based approaches to data-driven approaches, yielding learning-based coreference systems that perform much better than their hand-crafted counterparts. Many of these learning approaches follow the two step procedure as mentioned in section 3.1. In the classification step, those approaches can be differentiated from the following aspects (points a, b, c are given in Ng, 2005, point d is given by Luo et al., 2004):

(b) feature sets: Soon et al. (2001) define 12 surface level features which can be divided into four categories, lexical, grammatical, semantic and positional. Ng and Cardie (2002) extend the 12 features to a deeper set of 53 and the newly added features are based on common-sense knowledge and linguistic intuitions. Ng (2007) proposes another six semantic features, i.e., a semantic agreement feature, an ACE-specific semantic feature, a semantic similarity feature, a pattern-based feature, an anaphoricity feature and a coreferentiality feature. Yang and Su (2007) extract semantic relatedness from Wikipedia. Other papers covering the feature engineering include but not limit to Modjeska et al. (2003), Bean and Riloff (2004), Yang et al. (2005), Ponzetto and Strube (2006). We refer the readers to the original papers for details of those features.

(c) methods for creating training instances: McCarthy and Lehnert (1995) generate a positive instance by grouping each anaphoric mention paired with each of its coreferent antecedents, and a negative instance is created by pairing each mention with each of its preceding non-coreferent mentions. This method may produce quite large number of training instances. By contrast, Soon et al. (2001) create a smaller number of training instances, i.e., a positive instance is created for each anaphoric mention $em_i$ and its closest antecedent, $em_i$; and a negative instance is created for $em_i$ paired with each of the in-between mentions $em_{i+1}, ..., em_{j-1}$. The third method is given in (Ng and Cardie, 2002). A positive instance is created for each anaphoric mention and its most confident antecedent. For a non-pronominal mention, the most confident antecedent is its closest non-pronominal antecedent, and for pronouns, the most confident antecedent is its closest preceding antecedent. Negative instances are created as in Soon et al. (2001).

(d) mention-mention pair or mention-entity pair: some of the approaches compute the coreference likelihood of mention-mention pairs, e.g., Soon et al. (2001) and Ng and Cardie (2002). The other approaches compute the coreference likelihood of mention-entity pairs, e.g., Luo et al. (2004), Yang et al. (2008). It is worth noting that in the graph-based framework, since entity mentions are considered as nodes in the graph, coreference likelihood is computed for every mention-mention pair. The advantage of mention-mention pair is its computational simplicity: features are easy to compute over a pair of mentions, and its drawback is also obvious: the information outside the mention pair is ignored. Next, we show that the choice of computing mention-mention pair or mention-entity pair further affects the clustering step.

In the clustering step, we also have the following basic algorithms:

(a) closest-first clustering: each mention is grouped with its closest preceding referent as long as the mention-mention pair likelihood is above a given threshold (Soon et al., 2001).

(b) Best-first clustering: each mention is grouped with its preceding referent which produces the highest mention-mention pair likelihood. Since the most likely antecedent is chosen for each mention, best-first clustering may produce partitions with higher precision than closest-first clustering (Ng and Cardie, 2002).
(c) aggressive-merge clustering: each mention is grouped with all of its preceding referents. Such algorithm may produce overlapping clustering and may yield partitions with higher recalls.

While the above clustering algorithms are simple and reasonably successful, they suffer from a serious drawback:

- an instant decision (greedy style) is made when considering two mentions are coreferent or not, therefore, they makes no attempts to search through the space of all possible partitions which may lead to sub-optimal clustering (Luo et al., 2004; Ng, 2005)

To alleviate this problem, various approaches have been proposed, and quite some of the approaches have abandoned the scheme of two step procedure, e.g., LaSO framework (DaumeIII and Marcu, 2005), markov logic (Poon and Domingos, 2008). They are listed as follows but not limited to:

- **Luo et al. (2004)** use the Bell tree to represent the complete search space and each leaf node corresponds to a possible clustering outcome. However, since the search space becomes intractable as the number of mentions increases, Luo applies a heuristic beam search algorithm that will finally find the most probable partition, i.e., at each step of the search process, only the most promising nodes in the tree are expanded. This method cannot completely overcome the sub-optimal problem since the search is partial and heuristic driven.

- **Ng (2005)** developed 54 coreference resolution systems (by combinations of 3 classification algorithms, 3 clustering algorithms, 3 instance creation methods and 2 feature sets) and trained a global ranking model based on some partition related features. The ranking model is then used to produce the “optimal” coreference partition out of the 54 candidate partitions. However, their experiments only show modest improvements over the baseline systems using B-Cubed scoring measure which further imply that although his method can potentially expand the search space, the capability for searching the optimal one is still limited.

- **DaumeIII and Marcu (2005)** apply a Learning as Search Optimization (LaSO) framework that solves entity mention detection and coreference resolution in a simultaneous and joint manner. LaSO assumes there is a set of input structures $\mathcal{X}$ (in their case, documents), a set of output structures $\mathcal{Y}$ (in their case, documents with tagged entity mentions and coreference sets) and a search space $\mathcal{S}$ that connects $\mathcal{X}$ to $\mathcal{Y}$. The search space becomes even more intractable than Luo et al. (2004) because the number of candidate text spans (in their case chunk) is usually larger than the number of candidate entity mentions. Therefore, DaumeIII and Marcu also apply some heuristic driven strategy to avoid searching the whole space. The key idea is to perform search as normal until a point at which it becomes impossible to reach the correct solution.

- **Poon and Domingos (2008)** build a joint model based on markov logic (Richardson and Domingos, 2006) which is able to easily express relations among mentions, e.g., apposition and predicate nominals. In contrast to the pairwise mention-mention or mention-entity model, the joint model only takes some carefully designed first order predicates and clauses and performs joint inference among
mentions. Thus it runs in an unsupervised style, but can still achieve comparable performance to its supervised counterparts (even with significant better scores using MUC measure in their experiments, but according to our previous analysis, MUC measure has some drawbacks thus their results may be more convincing using B-Cubed or ECM measure). One of the key issues in their method is how to encode the linguistic and or world knowledge into predicates and clauses which turns out to be another skillful task. There are other joint models proposed so far, e.g., Non-Parametric Bayesian Models based on Dirichlet Processes (Haghighi and Klein 2007), Integer Linear Programming (Denis and Baldridge, 2007) and we refer readers to the original papers.

We now focus on discussing graph-based clustering methodology that has been successfully applied in solving entity coreference resolution. Most importantly, it also overcomes the drawbacks of two-step procedure. The major paper we will discuss is Nicolae and Nicolae (2006).

First, we present a summary of paper (Nicolae and Nicolae, 2006) as shown in Table 21, following the five part story in the graph-based clustering methodology.

<table>
<thead>
<tr>
<th>Modeling</th>
<th>Initially singleton entity mentions have been detected in a document.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- First group the entity mentions according to the entity type, and</td>
</tr>
<tr>
<td></td>
<td>start constructing a graph for each of the entity type.</td>
</tr>
<tr>
<td></td>
<td>- Use the entity mentions with entity type ( k ) as nodes in graph ( G_k ).</td>
</tr>
<tr>
<td></td>
<td>- Compute the coreference likelihood for any pairwise mention-mention</td>
</tr>
<tr>
<td></td>
<td>pairs in the graph. The likelihood is computed using a maximum</td>
</tr>
<tr>
<td></td>
<td>entropy model. Features applied in the model are shown in Table 22.</td>
</tr>
<tr>
<td></td>
<td>- Construct the full connected graph by connecting pairs of nodes</td>
</tr>
<tr>
<td></td>
<td>with edges, and the edge weight carries the coreference likelihood</td>
</tr>
<tr>
<td></td>
<td>computed in the previous step.</td>
</tr>
</tbody>
</table>

| Hypothesis                    | Entity mentions that corefer to each other must be clustered in a    |
|                               |   subgraph that contains more and better well-connected internal     |
|                               |   edges connecting the nodes in the subgraph than cutting edges     |
|                               |   connecting the nodes across subgraphs.                            |

| Measure                       | Minimum cut which is measured as the number of mentions that are    |
|                               |   correctly placed in their set (shown in Table 25).               |

| Algorithm                    | BESTCUT algorithm as shown in Table 26.                           |
| Evaluation                   | MUC measure and ECM measure                                       |

Table 21. Summary of paper (Nicolae and Nicolae, 2006)

<table>
<thead>
<tr>
<th>Category</th>
<th>Features</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical</td>
<td>exact_strm</td>
<td>1 if two mentions have the same spelling; 0 otherwise</td>
</tr>
<tr>
<td></td>
<td>left_subsm</td>
<td>1 if one mention is a left substring of the other; 0 otherwise</td>
</tr>
<tr>
<td></td>
<td>right_subsm</td>
<td>1 if one mention is a right substring of the other; 0 otherwise</td>
</tr>
<tr>
<td>Feature</td>
<td>Definition</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>acronym</td>
<td>1 if one mention is an acronym of the other; 0 otherwise</td>
<td></td>
</tr>
<tr>
<td>edit_dist</td>
<td>quantized editing distance between two mention strings</td>
<td></td>
</tr>
<tr>
<td>spell</td>
<td>pair of actual mention strings</td>
<td></td>
</tr>
<tr>
<td>ncd</td>
<td>number of different capitalized words in two mentions</td>
<td></td>
</tr>
<tr>
<td>head-match*</td>
<td>1 if the two heads are identical</td>
<td></td>
</tr>
<tr>
<td>type-pair*</td>
<td>for each mention: name-&gt;its type, noun-&gt;<em>NOUN</em>, pronoun-&gt;its spelling</td>
<td></td>
</tr>
<tr>
<td>name-alias*</td>
<td>1 if a mention is an alias of the other one</td>
<td></td>
</tr>
<tr>
<td>Distance</td>
<td>how many tokens two mentions are apart (quantized)</td>
<td></td>
</tr>
<tr>
<td>token_dist</td>
<td>how many sentences two mentions are apart (quantized)</td>
<td></td>
</tr>
<tr>
<td>sent_dist</td>
<td>how many sentences two mentions are apart (quantized)</td>
<td></td>
</tr>
<tr>
<td>gap_dist</td>
<td>how many mentions in between the two mentions in question (quantized)</td>
<td></td>
</tr>
<tr>
<td>Syntactic</td>
<td>POS-pair of two mention heads</td>
<td></td>
</tr>
<tr>
<td>POS_pair</td>
<td>POS-pair of two mention heads</td>
<td></td>
</tr>
<tr>
<td>apposition</td>
<td>1 if two mentions are appositive; 0 otherwise</td>
<td></td>
</tr>
<tr>
<td>same-governing</td>
<td>1 if both mentions are covered by the same type of node, e.g. NP, VP, PP</td>
<td></td>
</tr>
<tr>
<td>-category*</td>
<td>the parse tree path from $em_2$ to $em_1$</td>
<td></td>
</tr>
<tr>
<td>path*</td>
<td>1 if either mention collocates with a communication verb</td>
<td></td>
</tr>
<tr>
<td>coll-comm*</td>
<td>1 if either mention collocates with a communication verb</td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td>pair of (quantized) numbers, each counting how many times a mention string is seen</td>
<td></td>
</tr>
<tr>
<td>gender</td>
<td>pair of attributes of {female, male, neutral, unknown}</td>
<td></td>
</tr>
<tr>
<td>number</td>
<td>pair of attributes of {singular, plural, unknown}</td>
<td></td>
</tr>
<tr>
<td>possessive</td>
<td>1 if a pronoun is possessive; 0 otherwise</td>
<td></td>
</tr>
<tr>
<td>reflexive</td>
<td>1 if a pronoun is reflexive; 0 otherwise</td>
<td></td>
</tr>
<tr>
<td>grammatical</td>
<td>1 if the two mentions agree in gender and number</td>
<td></td>
</tr>
<tr>
<td>gn-agree*</td>
<td>1 if the two mentions agree in gender and number</td>
<td></td>
</tr>
</tbody>
</table>

Table 22. Features applied to compute coreference likelihood of entity mention-mention pairs (Luo et al. (2004)’s features + Nicolae and Nicolae (2006)’s 7 new features with names ending with star)

<table>
<thead>
<tr>
<th>Model</th>
<th>ECM-F(%)</th>
<th>#-features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>73.20 (±2.9)</td>
<td>171K</td>
</tr>
<tr>
<td>-syntax</td>
<td>72.6 (±2.5)</td>
<td>71K</td>
</tr>
<tr>
<td>-count</td>
<td>72.0 (±3.3)</td>
<td>70K</td>
</tr>
<tr>
<td>-dist</td>
<td>*66.2 (±3.9)</td>
<td>24K</td>
</tr>
<tr>
<td>-type/level</td>
<td>65.7 (±2.2)</td>
<td>5.4K</td>
</tr>
<tr>
<td>-spell</td>
<td>64.4 (±1.9)</td>
<td>39</td>
</tr>
</tbody>
</table>

Table 23. Impact of feature categories. Numbers after ± are the standard deviations, * indicates that the result is significantly (pair-wise t-test) different from the line above at
Luo et al. (2004)’s experiments (Table 23) show that the 39 basic features (e.g., string and substring match, acronym, edit distance and number of different capitalized words) can obtain 64.4 ECM F-score, distance features can bootstrap the performance most (from 66.2 to 72.0) and the final full-integrated features can obtain 73.2 ECM F-score.

<table>
<thead>
<tr>
<th>Model</th>
<th>ECM-F%</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>78.3</td>
</tr>
<tr>
<td>+grammatical</td>
<td>78.4</td>
</tr>
<tr>
<td>+lexical</td>
<td>83.1</td>
</tr>
<tr>
<td>+syntactic</td>
<td>85.1</td>
</tr>
</tbody>
</table>

Table 24. Impact of feature categories (Nicolae and Nicolae, 2006)

Nicolae and Nicolae (2006) used the features of Luo et al. (2004) as a baseline, and also evaluated their seven new features (in three categories, grammatical, lexical and syntactic) shown in Table 24. It shows that the three new features in lexical category bootstrap the performance most (from 78.4 to 83.1). It is worth noting that the baseline performance in Nicolae and Nicolae (2006) does not coincide with the full integrated feature performance in Luo et al. (2004), mostly probably because of the different algorithms they applied. But basically we can know what features are effective for the entity coreference resolution.

In contrast to the traditional minimum cut measure which is computed as the sum of the weights of the edges crossing a cut, the BESTCUT measure proposed by (Nicolae and Nicolae, 2006) is computed as the number of “correctly” placed nodes in the graph (Table 25). Therefore, the higher the cut weight, the better the cut is. A node is considered as “correctly” placed if the average weight of the edges connecting the node to the other nodes in its cluster (one side of the cut) is larger than the average weight of the edges connecting the node to the other nodes in the second cluster (the other side of the cut) or the maximum weight of the edge connecting the node to one of the nodes in its cluster is larger than the maximum weight of the edge connecting the node to one of the nodes in the other cluster. The final number of “correctly” placed nodes is then computed as the average of the number of “correctly” placed nodes in the two cases.

```plaintext
1  corrects_avg = corrects_max = 0
2  foreach m ∈ G.V
3      if m ∈ S.V then setm = S
4      else setm = T
5     if avg_n∈setm,V,n≠m weight(m,n) > avg_n∈G.V\setm.V weight(m,n)
6        then corrects_avg ++
7     if max_n∈setm,V,n≠m weight(m,n) > max_n∈G.V\setm.V weight(m,n)
8        then corrects_max ++
9  return (corrects_avg + corrects_max) / 2
```

\[ p = 0.05 \quad (\text{Luo et al., 2004}) \]
Table 25. BESTCUT measure of (Nicolae and Nicolae, 2006), computing the cut weight

<table>
<thead>
<tr>
<th>BESTCUT(Graph G_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 entities.clear()</td>
</tr>
<tr>
<td>2 queue.push(G_i)</td>
</tr>
<tr>
<td>3 while not queue.empty()</td>
</tr>
<tr>
<td>4 G ← queue.pop()</td>
</tr>
<tr>
<td>5 (S, T) ← ProposeCut(G)</td>
</tr>
<tr>
<td>6 if StopTheCut(G, S, T)</td>
</tr>
<tr>
<td>7 then</td>
</tr>
<tr>
<td>8 entities.add(NewEntity(G))</td>
</tr>
<tr>
<td>9 else</td>
</tr>
<tr>
<td>10 queue.push(S)</td>
</tr>
<tr>
<td>11 queue.push(T)</td>
</tr>
<tr>
<td>12 return entities</td>
</tr>
</tbody>
</table>

ProposeCut(Graph G)

1 while |G.V| > 1 |
2 (S, T) ← ProposeCutPhase(G) |
3 if the cut-weight of (S, T) is-larger 3 than the current best cut (S_b, T_b) |
4 then store the cut-of-the-phase (S, T) as the best cut (S_b, T_b) |
5 return (S_b, T_b) |

ProposeCutPhase(Graph G)

1 A ← \{G.V.first\} |
2 while |A| < |G.V| |
3 last ← the most tightly connected vertex |
4 add last to A |
5 store the cut-of-the-phase and shrink G by merging the last two added vertices |
6 return (G.V \ {last}, last) |

Table 26. BESTCUT algorithm proposed in (Nicolae and Nicolae, 2006+our revision)

The BESTCUT algorithm (Table 26) works as follows: the core of the algorithm is implemented in function BESTCUT, which accepts the constructed graph G_i (i indicates the entity type). The graph is first pushed into a queue. Then it enters a loop. If the queue is not empty, a graph (initially the graph G_i) is popped out which is denoted as G and a cut is proposed on G (implemented in function ProposeCutPhase). If the cut should be stopped (judged by function StopTheCut) which means G is well connected and breaking G is a bad thing, then G should be used to create a new entity (implemented by function NewEntity). If the cut leads to a reasonably good partitioning, then the two subgraphs aside the cut are pushed into the queue. Since the queue is not empty again, pop out a graph G and propose a cut on it. Therefore this procedure repeats until the queue becomes empty.

3 The original paper places “is-lighter” there, but since the cut weight has been refined by them, i.e., the larger the cut weight, the better the cut, thus, we consider “is-larger” is more intuitive. It is not a mistake, and the authors also explained the meaning of “is-lighter”.
Nicolae and Nicolae (2006) did not discuss much about the function ProposeCut which returns a cut of the graph $G$. However, since finding an “ideal” cut is a key issue of the whole algorithm, we feel that it is necessary to provide a detailed explanation. The algorithm for finding a cut is adapted from (Stoer and Wagner, 1997) which is known in the literature as maximum adjacency search or maximum cardinality search. The basis behind the algorithm is the theorem shown in Table 27.

Let $s$ and $t$ be two vertices of a graph $G$. Let $G/\{s, t\}$ be the graph obtained by merging $s$ and $t$. Then a minimum cut of $G$ can be obtained by taking the smaller of a minimum $s$-$t$-cut of $G$ and a minimum cut of $G/\{s, t\}$.

Table 27. Theorem for the best-cut algorithm (Stoer and Wagner, 1997)

In the theorem, a minimum $s$-$t$-cut of $G$ is implemented in function ProposeCutPhase and it works as follows: a subset $A$ starts with an arbitrary single vertex in the graph $(G, V, \text{first})$ and continues growing by adding a vertex that is most tightly connected with $A$ until the size of $A$ equals to the size of vertices in the graph. The most tightly connected vertex is defined as $z = \arg\max_{y \in A} w_a(A, y)$ where $w_a(A, y) = \sum_{x \in A} w(x, y)$. In the end, the last two added vertices are merged, i.e., the two vertices are replaced by a new vertex, and any edges from the two vertices to a remaining vertex are replaced by an edge weighted by the sum of the weights of the previous two edges. Edges joining the merged nodes are removed. The cut of $V$ that separates the last added vertex from the rest of the graph is called the cut-of-the-phase.

The procedure of ProposeCutPhase is repeated over and over again on the continuous shrinking graph, until the graph shrinks to only one node. During this iteration, we find the best cut computed by the measure in Table 25. It is worth noting that the original measure for best cut is computed as the sum of the weights of the edges crossing a cut, and the smaller the value, the better the cut. However, the authors have redefined the measure, and the larger the cut weight, the better the cut. Therefore, place “is-lighter” in the function of ProposeCut is counter-intuitive; therefore we revise it as “is-larger”.

The algorithm is guaranteed to find the best cut in the graph and we refer readers to (Stoer and Wagner, 1997) for detailed proof.

There is a final key function in the algorithm which is called StopTheCut and it determines whether cutting the set of mentions is better or worse than keeping the mentions together. This function is implemented by a leaning model and the authors provide great details in the paper, including how to create training instances (positive and negative instances) for the model and what features they select. We refer readers to the original paper for details.

The example below has been provided by the authors to illustrate how BESTCUT algorithm works, but we also use the same example to analyze whether the other algorithms can give correct results, e.g., the closest-first clustering and best-first clustering algorithms.

Example:

Mary$_1$ has a brother$_2$, John$_3$. The boy$_4$ is older than the girl$_5$.

In the example, there are five entity mentions which can be clustered into 2 entities, i.e., \{Mary$_1$, the girl$_5$\} and \{a brother$_2$, John$_3$, The boy$_4$\}.

For the BESTCUT algorithm, it works as follows: first, an initial graph is created as
shown in Figure 7. In the graph, the node number indicates the mention id, and the edge weights indicate the coreference likelihood for any pairs of mentions. However, if the likelihood is insignificant, the edge is removed, e.g., there is no edge between John, and the girl, or between Mary, and a brother. We also sort the vertices according to the node number, i.e., {1,2,3,4,5}. We start from node 1, and the most tight connected node is 5, and 4,3,2 respectively. Therefore, we obtain the first cut {1,5,4,3} and {2} as shown in Figure 8 (a), and the cut weight is 3, because mentions 1,2, 5 are correctly placed, and 3,4 are not. Then the graph shrinks by merging node 2 and 3. The second cut, third cut, and fourth cut are shown in Figure 8 (b), (c) and (d) respectively and the corresponding cut weights are 4, 5, 3.5 respectively. Therefore, the third cut is the best cut which has the largest cut weight. Because this is also the correct cut, a well learned model will probably declare against any further cuts. The final clustering then contains 2 clusters, {1,5} and {2,3,4}. The clustering is perfect.

We now apply closest-first clustering algorithm using the same example. We sort the mentions according to the node number first, and we start from the first mention. Since the coreference likelihood between mention 2 and mention 1 is below the threshold (there is no edge connecting them in the graph), mention 2 starts a new entity. For mention 3, it has connections with mention 1 and mention 2, but mention 2 is closer, so mention 3 is merged into the entity that already contains mention 2. For mention 4, the closest mention that has connection with it is mention 3, so mention 4 is also correctly merged. However, for mention 5, since the closest one that has connection with it is mention 4, so mention 5 is not correctly merged. The final clustering then contains 2 clusters, {1} and {2,3,4,5}. The clustering is not perfect.

We now analyze the best-first clustering algorithm. We also start from the first mention, and consider mention 2 which starts a new entity. Mention 3 has a best preceding mention which is mention 2, so mention 3 is correctly merged. The same applies to mention 4 and mention 4 is correctly merged with mention 3. For mention 5, the best preceding one is mention 1, so it can also be correctly merged with mention 1. The final clustering then contains 2 clusters, {1,5} and {2,3,4}. The clustering is also perfect.

In sum, both BESTCUT algorithm and best-first clustering algorithm work correctly in this example but closest-first algorithm does not produce good results.

Nicolae and Nicolae (2006) evaluated their algorithm in comparison with (Luo et al., 2004)'s Belltree and (Ng and Cardie, 2002)'s Link-Best algorithm using two measures: ECM F-measure and MUC F-measure. They tested each of the three algorithms (1) on the key mentions (annotated in the key files) (2) on the detected mentions (by their developed entity mention detection system) and (3) without any prior knowledge of the mention types. The results are shown in Table 28 and the conclusions are listed as follows:

- If the prior knowledge of mention types is known (either key or detected mentions), BESTCUT performs significantly better than the other two algorithms in the ECM F score and only slightly better in the MUC F score. But since MUC measure suffers from the two drawbacks (not considering single mentions and treating every error as equally important), ECM F measure is a more adequate measure.
- If BESTCUT has no information about the mention types, its performance is significantly below the other two algorithms in the ECM F score and MUC R score, but still significantly higher in the MUC P and MUC F score. This is consistent with
the fact that the first stage of BESTCUT algorithm divides the graph into subgraphs according to entity types.

![Figure 7. The initial graph](image)

![Figure 8. Cuts-of-the-phase](image)

<table>
<thead>
<tr>
<th>Clusterization algorithm</th>
<th>Mentions</th>
<th>ECM-F%</th>
<th>MUC score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MUC P%</td>
</tr>
<tr>
<td>BESTCUT</td>
<td>key</td>
<td>82.7</td>
<td>91.1</td>
</tr>
<tr>
<td></td>
<td>detected</td>
<td>73.0</td>
<td>88.3</td>
</tr>
<tr>
<td></td>
<td>undetected</td>
<td>41.2</td>
<td>52.0</td>
</tr>
<tr>
<td>Belltree (Luo et al., 2004)</td>
<td>key</td>
<td>77.9</td>
<td>88.5</td>
</tr>
<tr>
<td></td>
<td>detected</td>
<td>70.8</td>
<td>86.0</td>
</tr>
<tr>
<td></td>
<td>undetected</td>
<td>52.6</td>
<td>40.3</td>
</tr>
<tr>
<td>Link-Best (Ng and Cardie, 2002)</td>
<td>key</td>
<td>77.9</td>
<td>88.0</td>
</tr>
<tr>
<td></td>
<td>detected</td>
<td>70.7</td>
<td>85.1</td>
</tr>
</tbody>
</table>
Table 28. Evaluation of BESTCUT clustering algorithm (Nicolae and Nicolae, 2006)

To summarize the whole section of entity coreference resolution, we present the following highlighting points about graph-based clustering algorithm in comparison with other approaches that have been applied to entity coreference resolution:

- Graph is an elegant way to represent the problem and graph-based clustering algorithm solves the problem by optimizing a global objective function. But as any other approaches, the power of the algorithm also greatly relies on the information underlying the problem, e.g., the mention types, various syntax, semantic and pragmatic information for computing the coreference likelihood of mention pairs.

- There have not been any algorithms that can exactly ensure a global optimal clustering so far. As we have discussed earlier, solving a quality measure proposed so far is a NP-hard problem, thus any graph-based algorithms can only approximately produce the “optimal” results. (Luo et al., 2004)’s bell tree is a complete representation of the search space and the optimal clustering is located at a certain leaf in the tree, however, the heuristic searching algorithm proposed by them may still potentially lose the optimal solution. (Ng, 2005) obtained the “optimal” clustering by ranking candidate partitions generated by a set of coreference systems, however, the “optimal” one depends on the performance of the best coreference systems. The joint models based on markov logic (Poon and Domingos, 2008) also heavily rely on carefully designed rules, thus the “optimal” is not guaranteed.

- BESTCUT is only the first proposed graph-based clustering algorithm to solve entity coreference resolution and definitely not a final one. First, the minimum cut measure can be replaced by other quality measures; second, the running complexity of BESTCUT ($O(mn + n^2\log n)$ where $m$ is the number of edges, $n$ is the number of vertices) is high and as we have surveyed a set of other graph-based clustering algorithms, it is possible to extend the work by applying other algorithms with lower running complexity.

3.3 Event Coreference Resolution

The work of event coreference resolution can be traced back to MUC (Message Understanding Conference) Evaluations in the nineties of 20th century. Pioneering papers about the work include but may not be limited to (Humphreys et al., 1997) and (Bagga and Baldwin, 1998). We summarize the most important points of their work as follows:

- MUC defines a set of scenarios, e.g., management succession, resignation, election, espionage. Therefore, the events they studied are based on scenarios and the scenario names also define the event type names, e.g., management succession event, resignation event. However, a scenario can also contain sub events, e.g., a management succession event may involve two separate events of a company position being vacated by one person and then succeeded by another.

- Humphreys et al. (1997) proposed an inheritance-based semantic graph (ontology) in which objects, events, and attributes appear as nodes. The ontology is set up by
processing sentence by sentence and discourse entities (objects, events and attributes) are added as new nodes gradually. Coreference resolution is performed by comparing pairs of instances in which one instance is from current input sentence and the other is from an earlier processed sentence. The algorithm proceeds for each pair of instances by (1) checking semantic type consistency (2) checking attribute consistency and (3) computing a similarity score. The instances in the highest scoring pair (if there are any) are merged. The main issue in their algorithm is that the attribute consistency checking relies on manually defined rules, for example, two instances are not coreferential if they have incompatible times; two instances are not coreferential if different organizations or different management positions are involved in the management succession scenario.

- Bagga and Baldwin (1998) proposed a framework for cross-document entity coreference resolution and also adapted it to cross-document event coreference resolution. The whole procedure for cross-document entity coreference resolution can be split into three steps: (1) the coreference system produces entity coreference chains in each text; (2) with respect to an entity of interest, SentenceExtractor module extracts the sentences that contain the entity mentions on the entity chain and composes a summary for each text; (3) VSM (Vector Space Model)-Disambiguate module computes the similarities for pairs of summaries and summaries having similarity above a threshold are considered to be regarding the same entity. The version for cross-document event coreference resolution differs in that the SentenceExtractor module extracts the sentences that contain either the verb describing the event or one of its nominalizations. The main issue in their approach is that the verbs or their nominalizations that can describe an event should be known before extracting. Since the scenarios are only a few, it is not very difficult to list as many verbs as possible. However, it becomes an issue if we need to adapt the framework to more scenarios.

Succeeding the MUC Evaluation, the ACE (Automatic Content Extraction) program further advanced the research of event coreference resolution. Unlike the MUC events which are defined based on scenarios, ACE defines fine-grained event types and subtypes. An ACE event mention (an instance of an event) also has some important attributes, e.g., a distinguished trigger and a set of arguments. We refer readers back to section 3.1 since we have discussed great details in that section. We now review the papers regarding with ACE event coreference resolution. Ahn (2006) presented an event extraction system in which the component for event coreference resolution is located at the end of event extraction pipeline, however he did not mention great details about the event coreference resolution. Therefore we place greater emphasis on two recently published papers (Chen et al., 2009a; Chen and Ji, 2009b) on event coreference resolution, one is based on agglomerative clustering algorithm involving a pairwise event coreference model, and the other is based on spectral graph clustering algorithm.

The basic idea of agglomerative clustering (Chen et al., 2009a) is to start with singleton event mentions, traverse through each event mention (from left to right) and iteratively merge the active event mention into a prior established event or start the event mention as a new

---

4 Note that the entity here does not have the same meaning as we discussed earlier in entity coreference resolution, an entity here is the top node in the ontology, and has objects, events, and attributes as children.
event. Formally, let \( \{em_i; 1 \leq i \leq N\} \) be \( N \) event mentions in a document and the index \( i \) indicates the order it occurs in the document. Let \( e_j \) be the \( j^{th} \) event and \( f: i \to j \) be the map from event mention index \( i \) to event index \( j \). For each event mention index \( k(1 \leq k \leq N) \), let \( I_k = \{t: t = f(i) \text{ for } 1 \leq i \leq k - 1\} \) be the set of indices of partially-established events and \( E_k = \{e_t: t \in I_k\} \) be the set of partially-established events before the event mention \( em_k \) (note that \( E_1 = \emptyset \) and \( E_2 = \{[em_1]\} \)). We start the iteration from \( k = 2 \). At each iteration, find the event \( e_j \in E_k \) (\( j \) is the event index in \( E_k \)) such that

\[
e_j = \underset{e_t \in E_k}{\text{argmax}}(\text{coref}(e_t, em_k))
\]

where \( \text{coref}(\cdot, \cdot) \) is called pairwise (event-mention pair) coreference function that computes the coreference score between a prior event and the active event mention. If the highest score \( \text{coref}(e_j, em_k) \) is above a threshold \( \delta \), we merge \( em_k \) into event \( e_j \), otherwise, we start a new event and add it to \( E_k \). After \( N - 1 \) iterations, we resolve all the event coreferences in the document. The algorithm is shown in Table 29.

| Input: event mentions \( \{em_i; 1 \leq i \leq N\} \), coreference threshold \( \delta \) |
| Output: resolved events \( E_{N+1} \) |
| 1: Initialize \( E_1 = \emptyset \), \( E_2 = \{[em_1]\} \) |
| 2: for \( k = 2 \) to \( N \) { |
| 3: \( j = -1; \) prob = 0; |
| 4: foreach event \( e_t \in E_k \) { |
| 5: \( \text{if} (\text{coref}(e_t, em_k) > \text{prob}) \) { |
| 6: \( j = t; \) prob = \text{coref}(e_t, em_k); |
| 7: } |
| 8: } |
| 9: \( \text{if} (\text{prob} > \delta) \) { |
| 10: \( \text{Extend } e_j \text{ to } e'_j \text{ by merging } em_k \text{ into } e'_j; \) |
| 11: \( E_{k+1} = (E_k - \{e'_j\}) \cup \{e'_j\} \) |
| 12: } |
| 13: else |
| 14: \( E_{k+1} = E_k \cup \{em_k\} \) |
| 15: } |
| 16: return \( E_{N+1} \) |

Table 29. Agglomerative clustering algorithm for event coreference resolution

The agglomerative clustering looks intuitive and efficient with running complexity of \( O(N^2) \) where \( N \) is the number of event mentions, however, it suffers from a major drawback:

- Errors could be also agglomerative as the algorithm continues, i.e., if a wrong event mention is spuriously merged into a previous established event, it may probably hurt the event coreference model which computes the coreference value between an active event mention and the previously established event, and thus introduce more spurious event mentions later on.

By contrast, graph spectral clustering (Chen and Ji, 2009b) overcomes this problem since
all the coreference information has been encoded in the graph (event mentions are the vertices, and the edges carry the coreference likelihood between pairs of vertices). Due to the space limit in their paper, we provide a much detailed explanations in this review.

First, we also present a summary of paper (Chen and Ji, 2009b) as shown in Table 30, following the five part story in the graph-based clustering methodology.

<table>
<thead>
<tr>
<th>Modeling</th>
<th>Initially singleton event mentions have been detected in a document.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>● First group the event mentions according to the event type, and start constructing a graph for each of the event type.</td>
</tr>
<tr>
<td></td>
<td>● Use the event mentions with event type $k$ as nodes in graph $G_k$.</td>
</tr>
<tr>
<td></td>
<td>● Compute the coreference likelihood for any pairwise mention-mention pairs in the graph. Two methods were proposed in the paper to compute the likelihood, one is to use a maximum entropy model, and the other is to compute a formula.</td>
</tr>
<tr>
<td></td>
<td>● Construct the full connected graph by connecting pairs of nodes with edges, and the edge weight carries the coreference likelihood computed in the previous step.</td>
</tr>
</tbody>
</table>

| Hypothesis | Event mentions that corefer to each other must be clustered in a subgraph that contains more and better well-connected internal edges connecting the nodes in the subgraph than cutting edges connecting the nodes across subgraphs. |

| Measure | Normalized cut (Shi and Malik, 2000) as discussed in section 2.4.2 |
| Algorithm | Normalized spectral clustering algorithm I (Shi and Malik, 2000) |
| Evaluation | ECM F-measure |

Table 30. Summary of paper (Chen and Ji, 2009b)

We focus on discussing the two methods for computing the coreference likelihood of mention-mention pairs.

- Learning a maximum entropy model. The features applied in this model are listed in Table 31. In comparison with the features applied for entity coreference resolution, the category of distance shares some similarity, but the others are significantly different, most of which are related to trigger pairs and argument sets.

<table>
<thead>
<tr>
<th>Category</th>
<th>Features</th>
<th>Values ($em_1$: the first event mention, $em_2$: the second event mention)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexicon</td>
<td>type_subtype</td>
<td>pair of event type and subtype in $em_1$</td>
</tr>
<tr>
<td></td>
<td>trigger_pair</td>
<td>trigger pair of $em_1$ and $em_2$</td>
</tr>
<tr>
<td></td>
<td>pos_pair</td>
<td>part-of-speech pair of triggers of $em_1$ and $em_2$</td>
</tr>
<tr>
<td></td>
<td>nominal</td>
<td>1 if the trigger of EM2 is nominal</td>
</tr>
<tr>
<td></td>
<td>exact_match</td>
<td>1 if the spellings of triggers in $em_1$ and $em_2$ exactly match</td>
</tr>
<tr>
<td></td>
<td>stem_match</td>
<td>1 if the stems of triggers in $em_1$ and $em_2$ match</td>
</tr>
<tr>
<td></td>
<td>trigger_sim</td>
<td>quantized semantic similarity score (0-5) using WordNet resource</td>
</tr>
<tr>
<td>Distance</td>
<td>token_dist</td>
<td>how many tokens between triggers of $em_1$ and</td>
</tr>
</tbody>
</table>
Table 31. Features applied to compute coreference likelihood for event mention-mention pairs

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sentence_dist</td>
<td>how many sentences (e_{m1}) and (e_{m2}) are apart (quantized)</td>
</tr>
<tr>
<td>event_dist</td>
<td>how many event mentions in between (e_{m1}) and (e_{m2}) (quantized)</td>
</tr>
<tr>
<td>overlap_num, overlap_roles</td>
<td>overlap number of arguments and their roles (role and id exactly match)</td>
</tr>
<tr>
<td>prior_num, prior_roles</td>
<td>the number and the roles of arguments that only appear in (e_{m1})</td>
</tr>
<tr>
<td>act_num, act_roles</td>
<td>the number and the roles of arguments that only appear in (e_{m2})</td>
</tr>
<tr>
<td>coref_num</td>
<td>the number of arguments that corefer each other but have different roles</td>
</tr>
</tbody>
</table>

To illustrate how the features are computed, we use the example provided in (Chen and Ji, 2009b). Such details have been omitted in the paper. The example is shown in Table 32.

Table 32. An example for event coreference resolution

As an illustration, only event mentions with event type and subtype of (Conflict:Attack) are labeled in Table 32. There are 7 labeled event mentions in total and in each event mention, the trigger is surrounded by curly brackets, and arguments are underlined. A better structural representation of the 7 event mentions are listed in Table 33.

<table>
<thead>
<tr>
<th>EM1</th>
<th>Trigger: explosion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Arguments (ID: ROLE):</td>
</tr>
<tr>
<td></td>
<td>(E1-1: Place) a cafe at one of the capital's busiest intersections</td>
</tr>
<tr>
<td></td>
<td>(T1-1: Time-Within) Tuesday</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EM2</th>
<th>Trigger: explosion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Arguments:</td>
</tr>
<tr>
<td></td>
<td>(E2-1: Place) the restroom of the multistory Crocodile Cafe</td>
</tr>
<tr>
<td></td>
<td>(E3-1: Place) the commercial district of Kizilay</td>
</tr>
<tr>
<td></td>
<td>(T2-1: Time-Within) the morning rush hour</td>
</tr>
</tbody>
</table>
EM3 Trigger: blast
Arguments:
(E1-2: Place) the building

EM4 Trigger: explosion
Arguments:
(E4-1: Instrument) a bomb
(E1-3: Target) the site of the morning blast
(T3-1: Time-Within) morning

EM5 Trigger: explosion
Arguments: None

EM6 Trigger: exploded
Arguments:
(E5-1: Instrument) a bomb
(E6-1: Target) a McDonald's restaurant
(E7-1: Place) Istanbul

EM7 Trigger: bombings
Arguments:
(E8-1: Attacker) Radical leftist, Kurdish and Islamic groups
(E9-1: Place) the country
(T4-1: Time-Within) the past

As an example, we illustrate how to compute the feature vector of mention pair of EM1 and EM2. The results are listed in Table 34.

<table>
<thead>
<tr>
<th>Category</th>
<th>Features</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexicon</td>
<td>type_subtype</td>
<td>Conflict:Attack</td>
</tr>
<tr>
<td></td>
<td>trigger_pair</td>
<td>explosion: explosion</td>
</tr>
<tr>
<td></td>
<td>pos_pair</td>
<td>NN:NN</td>
</tr>
<tr>
<td></td>
<td>nominal</td>
<td>1 (because explosion is nominal)</td>
</tr>
<tr>
<td></td>
<td>exact_match</td>
<td>1 (because the spellings are exactly the same)</td>
</tr>
<tr>
<td></td>
<td>stem_match</td>
<td>1 (because the stems are exactly the same)</td>
</tr>
<tr>
<td></td>
<td>trigger_sim</td>
<td>5 (because they are semantically the same in WordNET)</td>
</tr>
<tr>
<td>Distance</td>
<td>token_dist</td>
<td>1 (116 tokens, but quantized by dividing by 100)</td>
</tr>
<tr>
<td></td>
<td>sentence_dist</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>event_dist</td>
<td>0</td>
</tr>
<tr>
<td>Arguments</td>
<td>overlap_num, overlap_</td>
<td>overlap_num= 0</td>
</tr>
<tr>
<td></td>
<td>roles</td>
<td></td>
</tr>
<tr>
<td></td>
<td>prior_num, prior_roles</td>
<td>prior_num=2, prior_roles= Place: Time-Within</td>
</tr>
<tr>
<td></td>
<td>act_num, act_roles</td>
<td>act_num=3, act_roles= Place: Place: Time-Within</td>
</tr>
<tr>
<td></td>
<td>coref_num</td>
<td>0</td>
</tr>
</tbody>
</table>

As an example, we illustrate how to compute the feature vector of mention pair of EM1 and EM2. The results are listed in Table 34.

There are two questions that have not been answered in (Chen and Ji, 2009b): (1) what is
the impact on the system performance for each feature category in Table 31; (2) does it significantly work better than agglomerative algorithm given in (Chen et al., 2009a). We provide answers as follows:

Table 35 shows that the baseline lexicon feature category (most features are about trigger pairs) can obtain 79.1% ECM F-score. The distance features contribute 3.6% improvements comparing with 5.8% improvements in entity coreference resolution (refer to Table 23). The argument features contribute another 1.9% improvements.

<table>
<thead>
<tr>
<th>Model</th>
<th>ECM-F%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexicon</td>
<td>79.1</td>
</tr>
<tr>
<td>+ Distance</td>
<td>82.7</td>
</tr>
<tr>
<td>+ Arguments</td>
<td>84.6</td>
</tr>
</tbody>
</table>

Table 35. Impact of feature categories

Table 36 shows the performance comparison between graph-based and agglomerative clustering algorithm. Given the ground-truth event mentions, Graph-based clustering algorithm is just slightly better than agglomerative clustering algorithm, 1% improvement in ECM F-score (significant at 95% confidence level using Wilcoxon signed rank tests), and 0.3% improvement in MUC F-score (significant at 70% confidence level using Wilcoxon signed rank tests). There are some issues that should be explained: (1) Chen et al. (2009a) use similar feature sets in their event coreference model with an additional feature set related with the four event attributes (modality, polarity, genericity, tense). The event attributes can actually be great helpful to distinguish the non-coreference from coreference, and encoding them as features also help to boost the performance. Unfortunately, Chen et al. (2009a) also show that it is a challenging task to obtain high accurate event attributes. For fair comparison, we have removed the event attribute feature set in agglomerative clustering algorithm. (2) To explain the poor performance score for detected mentions, Chen et al. (2009a) also show that the major bottleneck comes from the poor performance of system generated event mentions.

<table>
<thead>
<tr>
<th>Clusterization algorithm</th>
<th>Mentions</th>
<th>ECM-F%</th>
<th>MUC-F%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph-based</td>
<td>key</td>
<td>84.8</td>
<td>88.6</td>
</tr>
<tr>
<td></td>
<td>detected</td>
<td>56.2</td>
<td>50.4</td>
</tr>
<tr>
<td>Agglomerative</td>
<td>key</td>
<td>83.8</td>
<td>88.3</td>
</tr>
<tr>
<td></td>
<td>detected</td>
<td>55.4</td>
<td>49.2</td>
</tr>
</tbody>
</table>

Table 36. Performance comparison between graph-based and agglomerative clustering algorithm

- Computing a coreference formula

Chen and Ji (2009b) also proposed a formula to compute the coreference likelihood between two event mentions and it can also work surprisingly well. The idea behind constructing the formula is that the likelihood has close relations with both triggers and arguments. Using a corpus, we can obtain the statistics about event mention pairs, e.g., how
many pairs use exactly the same triggers in coreferring event mention pairs, how many
argument pairs whose ID and ROLE match in those coreferring event mention pairs.

The formula takes an exponential form \( e^{-\frac{1}{w_j^T+w_j^A}} \) in which \( w_j^T \) is a computed value
by comparing the trigger pair of event mention \( i \) and \( j \), and \( w_j^A \) is also a computed value
by comparing the pair of argument sets of event mention \( i \) and \( j \). Since the values for \( w_j^T \)
and \( w_j^A \) are greater than 0, the final coreference likelihood between event mention \( i \)
and \( j \) (the exponential form) falls in the range of 0 and 1.

The above formula considers that the comparison results from triggers and argument sets
are equally important. We can also propose a variance of the formula, such that \( w_j^T \) and \( w_j^A \)
are associated with biased weights, i.e., \( e^{-\frac{1}{\alpha w_j^T+(1-\alpha)w_j^A}} \) in which \( \alpha \) is a biased parameter
that falls in 0 and 1.

The great advantage of using the formula is that we do not need to train comprehensive
models (e.g., the maximal entropy model mentioned earlier) thus the running time can be
largely shortened without affecting the performance.

To summarize the whole section of event coreference resolution, we have the following
comments.

- Basically, the major techniques that have been successfully applied in entity
  coreference resolution can also be adapted to event coreference resolution. For
  example, Chen et al. (2009a) show that the major performance bottleneck of event
  coreference resolution for system generated resolution comes from the poor
  performance of system generated event mentions. An idea from entity coreference
  resolution is that a joint model can be developed so that event mention detection and
  coreference resolution can be done in a simultaneous and joint style, therefore, we
  can use techniques such as markov logic, Integrated Linear Programming.

- Absolutely, in most cases, an event is syntactically or structurally complex than an
  entity and an event contains lots more semantic meanings than an entity. It implies
  that we need to complete some refined work in order to capture the syntactic and
  semantic characteristics of an event. For example, neither Chen et al. (2009a) nor
  Chen and Ji (2009b) studied in-sentence event coreference, for some cases, two
  mentions in a sentence corefer and in many more cases, two mentions do not corefer.
  Therefore, refined models can be developed to handle in-sentence event coreference,
  in which parsing information may be useful.

- Although our work has not shown a significant advantage of graph-based clustering
  over agglomerative algorithm, we can still improve it by trying more graph-based
  clustering algorithms as have been surveyed in this report.

- Event coreference resolution and RTE (Recognizing Textual Entailment)\(^5\) task may
  complement each other.

---

\(^5\) RTE: given two text fragments, whether the meaning of one text is entailed (can be inferred) from another text
(Dagan et al., 2006).
4. Conclusions

The methodology of graph-based clustering surveyed in this paper can be applied in various research areas, not only for IE, but also for image processing, bioinformatics etc. Thus, we hope this literature survey can also “bridge” the interactions among different research communities. This literature survey has extensively discussed various quality measures, algorithms, and evaluation measures, thus it could also be used as a reference manual for researchers to harness the graph-based clustering methodology in their own problems.

References

[Measure and Algorithm]


**Evaluation**


[Coreference Resolution]


