Graph-based Clustering for Computational Linguistics: A Survey

Zheng Chen
The Graduate Center
The City University of New York
zchen1@gc.cuny.edu

Heng Ji
Queens College and The Graduate Center
The City University of New York
hengji@cs.qc.cuny.edu

Abstract

In this survey we overview graph-based clustering and its applications in computational linguistics. We summarize graph-based clustering as a five-part story: *hypothesis*, *modeling*, *measure*, *algorithm* and *evaluation*. We then survey three typical NLP problems in which graph-based clustering approaches have been successfully applied. Finally, we comment on the strengths and weaknesses of graph-based clustering and envision that graph-based clustering is a promising solution for some emerging NLP problems.

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 extensively investigating cluster properties, quality measures and various clustering algorithms by taking advantage of elegant mathematical structures built in graph theory. Practitioners have been investigating the graph clustering algorithms for specific applications and claiming their effectiveness by taking advantage of the underlying structure or other known characteristics of the data. Although graph-based clustering has gained increasing attentions from Computational Linguistic (CL) community (especially through the series of TextGraphs workshops), it is studied case by case and as far as we know, we have not seen much work on comparative study of various graph-based clustering algorithms for certain NLP problems. The major goal of this survey is to "bridge" the gap between theoretical aspect and practical aspect in graph-based clustering, especially for computational linguistics.

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 a graph can be partitioned into densely connected subgraphs that are sparsely connected to each other.
(2) Modeling. It deals with the problem of transforming data into a graph or modeling the real application as a graph.
(3) Measure. A quality measure is an objective function that rates the quality of a clustering.
(4) Algorithm. An algorithm is to exactly or approximately optimize the quality measure.
(5) Evaluation. Various metrics can be used to evaluate the performance of clustering by comparing with a "ground truth" clustering.

From the practical aspect, we focus on three typical NLP applications, including coreference resolution, word clustering and word sense disambiguation, in which graph-based clustering approaches have been successfully applied and achieved competitive performance.

2 Graph-based Clustering Methodology

We start with the basic clustering problem. Let $X = \{x_1, \ldots, x_N\}$ be a set of data points, $S = (s_{ij})_{i,j=1,\ldots,N}$ be the similarity matrix in which each element indicates the similarity $s_{ij} \geq 0$ between two data points x_i and x_j. A nice way to represent the data is to construct a graph on which each vertex represents a data point and the edge weight carries the similarity of two vertices. The clustering problem in graph perspective is then formulated as partitioning the graph into subgraphs such that the edges in the same subgraph have high weights and the edges between different subgraphs have low weights. In the next section, we define essential graph notation to facilitate discussions in the rest of this survey.

Proceedings of the 2010 Workshop on Graph-based Methods for Natural Language Processing, ACL 2010, pages 1–9,
Uppsala, Sweden, 16 July 2010.

2.1 Graph Notation

A graph is a triple $G=(V,E,W)$ where $V = \{v_1, \dots, v_N\}$ is a set of vertices, $E \subseteq V \times V$ is a set of edges, and $W = (w_{ij})_{i,j=1,\dots,N}$ 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.

In this survey we target at hard clustering problem which means we partition vertices of the graph into non-overlapping clusters, i.e., let $\mathcal{C} = (C_1, \dots, C_K)$ be a partition of V such that

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

2.2 Hypothesis

The hypothesis behind graph-based clustering can be stated in the following ways:

(1) The graph consists of dense subgraphs such that a dense subgraph contains more 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 subgraph will likely stay in the subgraph until many of its vertices have been visited (Dongen, 2000).
(3) Among all shortest paths between all pairs of vertices, links between different dense subgraphs are likely to be in many shortest paths (Dongen, 2000).

2.3 Modeling

Modeling addresses the problem of transforming the problem into graph structure, specifically, designating the meaning of vertices and edges in the graph, computing the edge weights for weighted graph, and constructing the graph. Luxburg (2006) stated three most common methods to construct a graph: *ε-neighborhood graph*, *k-nearest neighbor graph*, and *fully connected graph*. Luxburg analyzed different behaviors of the three graph construction methods, and stated that some graph-cluster algorithms (e.g., spectral clustering) can be quite sensitive to the choice of graphs and parameters ($\boldsymbol{\varepsilon}$ and $\boldsymbol{k}$). As a general recommendation, Luxburg suggested exploiting *k-nearest neighbor graph* as the first choice, which is less vulnerable to the choices of parameters than other graphs. Unfortunately, theoretical justifications on the choices of graphs and parameters do not exist and as a result, the problem has been ignored by practitioners.

2.4 Measure

A measure is an objective function that rates the quality of a clustering, thus called *quality measure*. By optimizing the quality measure, we can obtain the "optimal" clustering.

It is worth noting that *quality measure* should not be confused with *vertex similarity measure* where it is used to compute edge weights. Furthermore, we should distinguish *quality measure* from *evaluation measure* which will be discussed in section 2.6. The main difference is that cluster quality measure directly identifies a clustering that fulfills a desirable property while evaluation measure rates the quality of a clustering by comparing with a ground-truth clustering.

We summarize various quality measures in Table 1, from the basic density measures (*intra-cluster* and *inter-cluster*), to cut-based measures (*ratio cut*, *ncut*, *performance*, *expansion*, *conductance*, *bicriteria*), then to the latest proposed measure *modularity*. Each of the measures has strengths and weaknesses as commented in Table 1. Optimizing each of the measures is NP-hard. As a result, many efficient algorithms, which have been claimed to solve the optimal problem with polynomial-time complexity, yield sub-optimal clustering.

2.5 Algorithm

We categorize graph clustering algorithms into two major classes: divisive and agglomerative (Table 2). In the divisive clustering class, we categorize algorithms into several subclasses, namely, *cut-based*, *spectral clustering*, *multilevel*, *random walks*, *shortest path*. Divisive clustering follows top-down style and recursively splits a graph into subgraphs. In contrast, agglomerative clustering works bottom-up and iteratively merges singleton sets of vertices into subgraphs. The divisive and agglomerative algorithms are also called hierarchical since they produce multi-level clusterings, i.e., one clustering follows the other by refining (divisive) or coarsening (agglomerative). Most graph clustering algorithms ever proposed are divisive. We list the quality measure and the running complexity for each algorithm in Table 2.

Measures	Comments
intra-cluster density inter-cluster density	− Maximizing intra-cluster density is equivalent to minimizing inter-cluster density and vice versa − Drawback: both favor cutting small sets of isolated vertices in the graph (Shi and Malik, 2000)
ratio cut (Hagan and Kahng, 1992) ncut (Shi and Malik, 2000)	− Ratio cut is suitable for unweighted graph, and ncut is a better choice for weighted graph − Overcome the drawback of intra-cluster density or inter-cluster density − Drawback: both favor clusters with equal size
performance (Dongen, 2000; Brandes et al., 2003)	− Performance takes both intra-cluster density and inter-cluster density into considerations simultaneously
expansion, conductance, bicriteria (Kannan et al., 2000)	− Expansion is suitable for unweighted graph, and conductance is a better choice for weighted graph − Both expansion and conductance impose quality within clusters, but not inter-cluster quality; bicriteria takes both into considerations
modularity (Newman and Girvan,2004)	− Evaluates the quality of clustering with respect to a randomized graph − Drawbacks: (1) It requires global knowledge of the graph's topology, i.e., the number of edges. Clauset (2005) proposed an improved measure *Local Modularity*. (2) Resolution limit problem: it fails to identify clusters smaller than a certain scale. Ruan and Zhang (2008) proposed an improved measure *HQcut*. (3) It fails to distinguish good from bad clustering between different graphs with the same modularity value. Chen et al. (2009) proposed an improved measure *Max-Min Modularity*

Table 1. Summary of Quality Measures

Category		Algorithms	optimized measure	running complexity
divisive	cut-based	*Kernighan-Lin algorithm* (Kernighan and Lin, 1970)	*intercluster*	$O(\|V\|^3)$
		cut-clustering algorithm (Flake et al., 2003)	*bicriteria*	$O(\|V\|)$
	spectral	*unnormalized spectral clustering* (Luxburg, 2006)	*ratiocut*	$O(\|V\|\|E\|)$
		normalized spectral clustering I (Luxburg, 2006; Shi and Malik, 2000)	*ncut*	$O(\|V\|\|E\|)$
		normalized spectral clustering II (Luxburg, 2006; Ng, 2002)	*ncut*	$O(\|V\|\|E\|)$
		iterative conductance cutting (ICC) *(*Kannan et al.,2000)	*conductance*	$O(\|V\|\|E\|)$
		geometric MST clustering (GMC) (Brandes et al., 2007)	*pluggable(any quality measure)*	$O(\|V\|\|E\|)$
		modularity oriented (White and Smyth,2005)	*modularity*	$O(\|V\|\|E\|)$
	multilevel	*multilevel recursive bisection* (Karypis and Kumar, 1999)	*intercluster*	$O(\|V\|logK)$
		multilevel K-way partitioning (Karypis and Kumar, 1999)	*intercluster*	$O(\|V\| + KlogK)$
	random	*Markov Clustering Algorithm (MCL)* (Dongen, 2000)	*performance*	$O(m^2\|V\|)$
	shortest path	*betweenness* (Girvan and Newman, 2003)	*modularity*	$O(\|V\|\|E\|^2)$
		information centrality (Fortunato et al., 2004)	*modularity*	$O(\|V\|\|E\|^3)$
agglomerative		*modularity oriented* (Newman, 2004)	*modularity*	$O(\|V\|\|E\|)$

Table 2. Summary of Graph-based Clustering Algorithms ($|V|$: the number of vertices, $|E|$: the number of edges, K: the number of clusters, m: the number of resources allocated for each vertex)

The first set of algorithms (*cut-based*) is associated with *max-flow min-cut* theorem (Ford and Fulkerson, 1956) which states that "the value of the maximum flow is equal to the cost of the minimum cut". One of the earliest algorithm, *Kernighan-Lin* algorithm (Kernighan and Lin, 1970) splits the graph by performing recursive bisection (split into two parts at a time), aiming to minimize inter-cluster density (cut size). The high complexity of the algorithm ($O(|V|^3)$ makes it less competitive in real applications. Flake et al. (2003) proposed a cut-clustering algorithm which optimizes the bicriterion measure and the complexity is proportional to the number of clusters K using a heuristic, thus the algorithm is competitive in practice.

The second set of algorithms is based on spectral graph theory with Laplacian matrix as the mathematical tool. The connection between clustering and spectrum of Laplacian matrix (L) basically lies in the following important proposition: the multiplicity k of the eigenvalue 0 of L equals to the number of connected components in the graph. Luxburg (2006) and Abney (2007) presented a comprehensive tutorial on spectral clustering. Luxburg (2006) discussed three forms of Laplacian matrices (one unnormalized form and two normalized forms) and their three corresponding spectral clustering algorithms (unnormalized, normalized I and normalized II). Unnormalized clustering aims to optimize *ratiocut* measure while normalized clustering aims to optimize *ncut* measure (Shi and Malik, 2000), thus spectral clustering actually relates with cut-based clustering. The success of spectral clustering is mainly based on the fact that it does not make strong assumptions on the form of the clusters and can solve very general problems like intertwined spirals which k-means clustering handles much worse. Unfortunately, spectral clustering could be unstable under different choices of graphs and parameters as mentioned in section 2.3. Luxburg et al. (2005) compared unnormalized clustering with normalized version and proved that normalized version always converges to a sensible limit clustering while for unnormalized case the same only holds under strong additional assumptions which are not always satisfied. The running complexity of spectral clustering equals to the complexity of computing the eigenvectors of Laplacian matrix which is $O(|V|^3)$. However, when the graph is sparse, the complexity is reduced to $O(|V||E|)$ by applying efficient *Lanczos* algorithm.

The third set of algorithms is based on multilevel graph partitioning paradigm (Karypis and Kumar, 1999) which consists of three phases: coarsening phase, initial partitioning phase and refinement phase. Two approaches have been developed in this category, one is *multilevel recursive bisection* which recursively splits into two parts by performing multilevel paradigm with complexity of $O(|V|logK)$; the other is *multilevel K-way partitioning* which performs coarsening and refinement only once and directly partitions the graph into K clusters with complexity of $O(|V| + KlogK)$. The latter approach is superior to the former one for less running complexity and comparable (sometimes better) clustering quality.

The fourth set of algorithms is based on the second interpretation of the hypothesis in section 2.2, i.e., a random walk is likely to visit many vertices in a cluster before moving to the other cluster. An outstanding approach in this category is presented in Dogen (2000), named Markov clustering algorithm (MCL). The algorithm iteratively applies two operators (expansion and inflation) by matrix computation until convergence. Expansion operator simulates spreading of random walks and inflation models demotion of inter-cluster walks; the sequence matrix computation results in eliminating inter-cluster interactions and leaving only intra-cluster components. The complexity of MCL is $O(m^2|V|)$ where m is the number of resources allocated for each vertex. A key point of random walk is that it is actually linked to spectral clustering (Luxburg, 2006), e.g., *ncut* can be expressed in terms of transition probabilities and optimizing *ncut* can be achieved by computing the stationary distribution of a random walk in the graph.

The final set of algorithms in divisive category is based on the third interpretation of the hypothesis in section 2.2, i.e., the links between clusters are likely to be in the shortest paths. Girvan and Newman (2003) proposed the concept of edge *betweenness* which is the number of shortest paths connecting any pair of vertices that pass through the edge. Their algorithm iteratively removes one of the edges with the highest *betweenness*. The complexity of the algorithm is $O(|V||E|^2)$. Instead of *betweenness*, Fortunato et al. (2004) used *information centrality* for each edge and stated that it performs better than *betweenness* but with a higher complexity of $O(|V||E|^3)$.

The agglomerative category contains much fewer algorithms. Newman (2004) proposed an

algorithm that starts each vertex as singletons, and then iteratively merges clusters together in pairs, choosing the join that results in the greatest increase (or smallest decrease) in *modularity* score. The algorithm converges if there is only cluster left in the graph, then from the clustering hierarchy, we choose the clustering with maximum *modularity*. The complexity of the algorithm is $O(|V||E|)$.

The algorithms we surveyed in this section are by no means comprehensive as the field is long-standing and still evolving rapidly. We also refer readers to other informative references, e.g., Schaeffer (2007), Brandes et al. (2007) and Newman (2004).

A natural question arises: "which algorithm should we choose?" A general answer to this question is that no algorithm is a panacea. First, as we mentioned earlier, a clustering algorithm is usually proposed to optimize some quality measure, therefore, it is not fair to compare an algorithm that favors one measure with the other one that favors some other measure. Second, there is not a perfect measure that captures the full characteristics of cluster structures; therefore a perfect algorithm does not exist. Third, there is no definition for so called "best clustering". The "best" depends on applications, data characteristics, and granularity.

2.6 Evaluation

We discussed various quality measures in section 2.4, however, a clustering optimizing some quality measure does not necessarily translate into effectiveness in real applications with respect to the ground truth clustering and thus an evaluation measure plays the role of evaluating how well the clustering matches the gold standard. Two questions arise: (1) what constraints (properties, criteria) should an ideal evaluation measure satisfy? (2) Do the evaluation measures ever proposed satisfy the constraints?

For the first question, there have been several attempts on it: Dom (2001) developed a parametric technique for describing the quality of a clustering and proposed five "desirable properties" based on the parameters; Meila (2003) listed 12 properties associated with the proposed entropy measure; Amigo et al. (2008) proposed four constraints including homogeneity, completeness, rag bag, and cluster size vs. quantity. A parallel comparison shows that the four constraints proposed by Amigo et al. (2008) have advantages over the constraints proposed in the other two papers, for one reason, the four constraints can describe all the important constraints in Dom (2001) and Meila (2003), but the reverse does not hold; for the other reason, the four constraints can be formally verified for each evaluation measure, but it is not true for the constraints in Dom (2001).

Table 3 lists the evaluation measures ever proposed (including those discussed in Amigo et al., 2008 and some other measures known for coreference resolution). To answer the second question proposed in this section, we conclude the findings in Amigo et al. (2008) plus our new findings about MUC and CEAF as follows: (1) all the measures except B-Cubed fail the rag bag constraint and only B-Cubed measure can satisfy all the four constraints; (2) two entropy based measures (VI and V) and MUC only fail the rag bag constraint; (3) all the measures in set mapping category fail completeness constraint (4) all the measures in pair counting category fail cluster size vs. quantity constraint; (5) CEAF, unfortunately, fails homogeneity, completeness, rag bag constraints.

Category	Evaluation Measures
set mapping	purity, inverse purity, F-measure
pair counting	rand index, Jaccard Coefficient, Folks and Mallows FM
entropy	entropy, mutual information, VI, V
editing distance	editing distance
coreference resolution	MUC (Vilain et al.,1995), B-Cubed (Bagga and Baldwin, 1998), CEAF (Luo, 2005)

Table 3. Summary of Evaluation Measures

3 Applying Graph Clustering to NLP

A variety of structures in NLP can be naturally represented as graphs, e.g., co-occurrence graphs, coreference graphs, word/sentence/ document graphs. In recent years, there have been an increasing amount of interests in applying graph-based clustering to some NLP problems, e.g., document clustering (Zhong and Ghosh, 2004), summarization (Zha, 2002), coreference resolution (Nicolae and Nicolae, 2006), word sense disambiguation (Dorow and Widdows, 2003; Véronis, 2004; Agirre et al., 2007), word clustering (Matsuo et al., 2006; Biemann, 2006). Many authors chose one or two their favorite graph clustering algorithms and claimed the effectiveness by comparing with supervised algorithms (which need expensive annotations) or other non-

graph clustering algorithms. As far as we know, there is not much work on the comparative study of various graph-based clustering algorithms for certain NLP problems. As mentioned at the end of section 2.5, there is not a graph clustering algorithm that is effective for all applications. However, it is interesting to find out, for a specific NLP problem, if graph clustering methods can be applied, (1) how the parameters in the graph model affects the performance? (2) Does the NLP problem favor some quality measure and some graph clustering algorithm rather than the others? Unfortunately, this survey neither provides answers for these questions; instead, we overview a few NLP case studies in which some graph-based clustering methods have been successfully applied.

3.1 Coreference Resolution

Coreference resolution is typically defined as the problem of partitioning a set of *mentions* into *entities*. 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. The approaches to solving coreference resolution have shifted from earlier linguistics-based (rely on domain knowledge and handcrafted rules) to machine-learning based approaches. Elango (2005) and Chen (2010) presented a comprehensive survey on this topic. One of the most prevalent approaches for coreference resolution is to follow a two-step procedure: (1) a classification step that computes how likely one mention corefers with the other and (2) a clustering step that groups the mentions into clusters such that all mentions in a cluster refer to the same entity. In the past years, NLP researchers have explored and enriched this methodogy from various directions (either in classification or clustering step). Unfortunately, most of the proposed clustering algorithms, e.g., closest-first clustering (Soon et al., 2001), best-first clustering (Ng and Cardie, 2002), suffer from a drawback: an instant decision is made (in greedy style) when considering two mentions are coreferent or not, therefore, the algorithm makes no attempt to search through the space of all possible clusterings, which results in a sub-optimal clustering (Luo et al., 2004). Various approaches have been proposed to alleviate this problem, of which graph clustering methodology is one of the most promising solutions.

The problem of coreference resolution can be modeled as a graph such that the vertex represents a mention, and the edge weight carries the coreference likelihood between two mentions. Nicolae and Nicolae (2006) proposed a new quality measure named BESTCUT which is to optimize the sum of "correctly" placed vertices in the graph. The BESTCUT algorithm works by performing recursive bisection (similar to *Kernighan-Lin* algorithm) and in each iteration, it searches the best cut that leads to partition into halves. They compared BESTCUT algorithm with (Luo et al., 2004)'s Belltree and (Ng and Cardie, 2002)'s Link-Best algorithm and showed that using ground-truth entities, BESTCUT outperforms the other two with statistical significance (4.8% improvement over Belltree and Link-Best algorithm in ECM F-measure). Nevertheless, we believe that the BESTCUT algorithm is not the only choice and the running complexity of BESTCUT,$O(|V||E| + |V|^2 log|V|)$, is not competitive, thus could be improved by other graph clustering algorithms.

Chen and Ji (2009a) applied normalized spectral algorithm to conduct event coreference resolution: partitioning a set of *mentions* into *events.* 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). A graph is similarly constructed as in entity coreference resolution except that it involves quite different feature engineering (most features are related with event *trigger* and *arguments*). The graph clustering approach yields competitive results by comparing with an agglomerative clustering algorithm proposed in (Chen et al., 2009b), unfortunately, a scientific comparison among the algorithms remains unexplored.

3.2 Word Clustering

Word clustering is a problem defined as clustering a set of words (e.g., nouns, verbs) into groups so that similar words are in the same cluster. Word clustering is a major technique that can benefit many NLP tasks, e.g., thesaurus construction, text classification, and word sense disambiguation. Word clustering can be solved by following a two-step procedure: (1) classification step by representing each word as a feature vector and computing the similarity of two words; (2) clustering step which applies some clustering algorithm, e.g., single-link clustering, complete-link clustering, average-link clustering, such that similar words are grouped together.

Matsuo et al. (2006) presented a graph cluster-

ing algorithm for word clustering based on word similarity measures by web counts. A word co-occurrence graph is constructed in which the vertex represents a word, and the edge weight is computed by applying some similarity measure (e.g., PMI, χ^2) on a co-occurrence matrix, which is the result of querying a pair of words to a search engine. Then an agglomerative graph clustering algorithm (Newman, 2004), which is surveyed in section 2.5, is applied. They showed that the similarity measure χ^2 performs better than PMI, for one reason, PMI performs worse when a word group contains rare or frequent words, for the other reason, PMI is sensitive to web output inconsistency, e.g., the web count of w_1 is below the web count of $w_1 AND w_2$ in extreme case. They also showed that their graph clustering algorithm outperforms average-link agglomerative clustering by almost 32% using χ^2 similarity measure. The concern of their approach is the running complexity for constructing co-occurrence matrix, i.e., for n words, $O(n^2)$ queries are required which is intractable for a large graph.

Ichioka and Fukumoto (2008) applied similar approach as Matsuo et al. (2006) for Japanese Onomatopoetic word clustering, and showed that the approach outperforms k-means clustering by 16.2%.

3.3 Word Sense Disambiguation (WSD)

Word sense disambiguation is the problem of identifying which sense of a word (meaning) is conveyed in the context of a sentence, when the word is polysemic. In contrast to supervised WSD which relies on pre-defined list of senses from dictionaries, unsupervised WSD induces word senses directly from the corpus. Among those unsupervised WSD algorithms, graph-based clustering algorithms have been found competitive with supervised methods, and in many cases outperform most vector-based clustering methods.

Dorow and Widdows (2003) built a co-occurrence graph in which each node represents a noun and two nodes have an edge between them if they co-occur more than a given threshold. They then applied Markov Clustering algorithm (MCL) which is surveyed in section 2.5, but cleverly circumvent the problem of choosing the right parameters. Their algorithm not only recognizes senses of polysemic words, but also provides high-level readable cluster name for each sense. Unfortunately, they neither discussed further how to identify the sense of a word in a given context, nor compared their algorithm with other algorithms by conducting experiments.

Véronis (2004) proposed a graph based model named *HyperLex* based on the small-world properties of co-occurrence graphs. Detecting the different senses (uses) of a word reduces to isolating the high-density components (hubs) in the co-occurrence graph. Those hubs are then used to perform WSD. To obtain the hubs, *HyperLex* finds the vertex with highest relative frequency in the graph at each iteration and if it meets some criteria, it is selected as a hub. Agirre (2007) proposed another method based on *PageRank* for finding hubs. *HyperLex* can detect low-frequency senses (as low as 1%) and most importantly, it offers an excellent precision (97% compared to 73% for baseline). Agirre (2007) further conducted extensive experiments by comparing the two graph based models (*HyperLex* and *PageRank*) with other supervised and non-supervised graph methods and concluded that graph based methods perform close to supervised systems in the lexical sample task and yield the second-best WSD systems for the Senseval-3 all-words task.

4 Conclusions

In this survey, we organize the sparse related literature of graph clustering into a structured presentation and summarize the topic as a five part story, namely, hypothesis, modeling, measure, algorithm, and evaluation. The hypothesis serves as a basis for the whole graph clustering methodology, quality measures and graph clustering algorithms construct the backbone of the methodology, modeling acts as the interface between the real application and the methodology, and evaluation deals with utility. We also survey several typical NLP problems, in which graph-based clustering approaches have been successfully applied.

We have the following final comments on the strengths and weaknesses of graph clustering approaches:

(1) Graph is an elegant data structure that can model many real applications with solid mathematical foundations including spectral theory, Markov stochastic process.

(2) Unlike many other clustering algorithms which act greedily towards the final clustering and thus may miss the optimal clustering, graph clustering transforms the clustering problem into optimizing some quality measure. Unfortunately, those optimization problems are NP-Hard, thus, all proposed graph

clustering algorithms only approximately yield "optimal" clustering.

(3) Graph clustering algorithms have been criticized for low speed when working on large scale graph (with millions of vertices). This may not be true since new graph clustering algorithms have been proposed, e.g., the multilevel graph clustering algorithm (Karypis and Kumar, 1999) can partition a graph with one million vertices into 256 clusters in a few seconds on current generation workstations and PCs. Nevertheless, scalability problem of graph clustering algorithm still needs to be explored which is becoming more important in social network study.

We envision that graph clustering methods can lead to promising solutions in the following emerging NLP problems:

(1) Detection of new entity types, relation types and event types (IE area). For example, the eight event types defined in the ACE[1] program may not be enough for wider usage and more event types can be induced by graph clustering on verbs.

(2) Web people search (IR area). The main issue in web people search is the ambiguity of the person name. Thus by extracting attributes (e.g., attended schools, spouse, children, friends) from returned web pages, constructing person graphs (involving those attributes) and applying graph clustering, we are optimistic to achieve a better person search engine.

Acknowledgments

This work was supported by the U.S. National Science Foundation Faculty Early Career Development (CAREER) Award under Grant IIS-0953149, the U.S. Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, Google, Inc., CUNY Research Enhancement Program, Faculty Publication Program and GRTI Program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

[1] http://www.nist.gov/speech/tests/ace/

References

A. Bagga and B. Baldwin.1998. Algorithms for scoring coreference chains. *Proc. The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference.*

A. Clauset. 2005. Finding local community structure in networks. *Physical Review E*, 72:026132.

B. Dom. 2001. An information-theoretic external cluster-validity measure. *IBM Research Report*.

B. Dorow, D. Widdows. 2003. Discovering corpus-specific word-senses. In *Proc. EACL*.

B. W. Kernighan and S. Lin.1970. An efficient heuristic procedur for partitioning graphs. Bell Syst. Techn. J.,Vol. 49, No. 2, pp. 291–307.

C. Biemann. 2006. Chinese Whispers - an Efficient-Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proc. of the HLT-NAACL-06 Workshop on Textgraphs-06*.

C. Nicolae and G. Nicolae. 2006. Bestcut: A graph algorithm for coreference resolution. In *EMNLP*, pages 275–283, Sydney, Australia.

E. Agirre, D. Martinez, O.L. de Lacalle and A.Soroa. 2007. Two graph-based algorithms for state-of-the-art WSD. In *Proc. EMNLP*.

E. Amigo, J. Gonzalo, J. Artiles and F. Verdejo. 2008. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*.

E. Terra and C. L. A. Clarke. Frequency Estimates for Statistical Word Similarity Measures. In *Proc. HLT/NAACL 2003*.

G. Karypis and V. Kumar. 1999. Multilevel algorithms for multiconstraint graph partitioning. in *Proceedings of the 36th ACM/IEEE conference on Design automation conference*, (New Orleans, Louisiana), pp. 343 – 348.

G. W. Flake, R. E. Tarjan and K. Tsioutsiouliklis. 2003. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408.

H. Zha.2002. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proc. of SIGIR2002*, pp. 113-120.

J. Chen, O. R. Zaïane, R. Goebel. 2009. Detecting Communities in Social Networks Using Max-Min Modularity. *SDM 2009*: 978-989.

J. Ruan and W. Zhang.2008. Identifying network communities with a high resolution. Physical Review E, 77:016104.

J. Shi and J. Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888-905.

J. Véronis. 2004. HyperLex: Lexical Cartography for Information Retrieval. *Computer Speech & Language* 18(3).

K. Ichioka and F. Fukumoto. 2008. Graph-based clustering for semantic classification of onomatopoetic words. In *Proc. of the 3rd Textgraphs Workshop on Graph-based Algorithms for Natural Language Processing*.

L. Hagen and A. B. Kahng. 1992. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions Computer-Aided Design*, Santa Clara CA, 422-427.

L. R. Ford, D. R. Fulkerson. 1956. Maximal flow through a network. *Canadian Journal of Mathematics* 8: 399–404.

M. E. J. Newman. 2004. Detecting community structure in networks. Eur. Phys. J. B, 38, 321–330.

M. E. J. Newman. 2004. Fast algorithm for detecting community structure in networks. *Phys Rev E.* 69, 2004.

M. E. J. Newman and M. Girvan. 2004. Finding and evaluating community structure in networks. *Phys. Rev. E* 69,026113.

M. Girvan and M. E. J. Newman. 2002. Community structure in social and biological networks. *Proc. Natl. Acad. Sci*. USA 99, 7821-7826.

M. Meila. 2003. Comparing clusterings. In *Proceedings of COLT03*.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly and L. Hirschman.1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*.

P. Elango. 2005. Coreference Resolution: A Survey. *Technical Report*, University of Wisconsin Madison.

R. Kannan, S. Vempala, and A. Vetta. 2000. On clusterings:good, bad and spectral. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*.

S. Abney. 2007. Semi-supervised Learning for Computational Linguistics, Chapman and Hall.

S. E. Schaeffer. 2007. Graph clustering. *Computer Science Review*, 1(1):27–64.

S. Fortunato, V. Latora, and M. Marchiori. 2004. A Method to Find Community Structures Based on Information Centrality. *Phys Rev E*.70, 056104.

S. van Dongen. 2000. Graph Clustering by Flow Simulation. *PhD thesis*, University of Utrecht.

S. White and P. Smyth. 2005. A spectral clustering approach to finding communities in graphs. In *SIAM International Conference on Data Mining*.

U. Brandes, M. Gaertler, and D. Wagner. 2003. Experiments on graph clustering algorithms. *Proc. 11th European Symp. Algorithms, LNCS* 2832:568-579.

U. Brandes, M. Gaertler, and D.Wagner. 2007. Engineering graph clustering: Models and experimental evaluation. *J. Exp. Algorithmics*, 12:1.1.

U. Luxburg, O. Bousquet, M. Belkin. 2005. Limits of spectral clustering. In *L. K. Saul, Y. Weiss and L. Bottou (Eds.), Advances in neural information processing systems* 17. Cambridge, MA: MIT Press.

U. Luxburg.2006. A tutorial on spectral clustering. Technical Report 149, *Max Plank Institute for Biological Cybernetics*.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proc. Of the ACL*, pages 104–111.

W. M. Soon, H. T. Ng and D. Lim.2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

X. Luo. 2005. On coreference resolution performance metrics. *Proc. of HLT-EMNLP*.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell Tree. In *Proc. of ACL-04*, pp.136–143.

Y. Matsuo, T. Sakaki, K. Uchiyama, and M. Ishizuka. 2006. Graph-based word clustering using web search engine. In *Proc. of EMNLP 2006*.

Z. Chen and H. Ji. 2009a. Graph-based Event Coreference Resolution. In *Proc. ACL-IJCNLP 2009 workshop on TextGraphs-4: Graph-based Methods for Natural Language Processing*.

Z. Chen, H. Ji, R. Haralick. 2009b. A Pairwise Coreference Model, Feature Impact and Evaluation for Event Coreference Resolution. In *Proc. RANLP 2009 workshop on Events in Emerging Text Types*.

Z. Chen. 2010. Graph-based Clustering and its Application in Coreference Resolution. *Technical Report*, the Graduate Center, the City University of New York.