Bio-inspired constrained clustering: A case study on aspect-based sentiment analysis

by

Mohammed Qasem

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Doctor of Philosophy

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
February 2018

Thesis advisor

Parimala Thulasiraman

Author

Mohammed Qasem

Bio-inspired constrained clustering: A case study on aspect-based sentiment analysis

Abstract

Clustering is an important problem in the era of big data. Exact algorithmic clustering approaches are not affordable for many real-world applications (RWA), requiring innovative, approximation algorithms. Among them are bio or nature-inspired techniques such as *ant brood clustering algorithm* (ACA) inspired by how real ants brood sort their nests.

ACA's mathematical model assumes a static radius of perception which is not adaptable to RWA. I address this issue by developing an adaptive clustering algorithm, called *ACA with Adaptive Radius (ACA-AR)* using kernel density estimation, a non-parametric statistical model, to measure average dissimilarity of data objects in ants neighborhood. I extend this algorithm to a search-based semi-supervised constrained clustering algorithm (CACA-AR) that incorporates supervisory information to guide the clustering algorithm towards solutions where constraints are minimally violated. I evaluate the accuracy of CACA-AR on benchmark datasets and provide a feasibility study on one RWA, aspect based sentiment analysis. The F1-score results show that CACA-AR outperforms baseline techniques, multi-class logistic regression and lexicon based approaches by 20%.

Contents

List of Figures

List of Tables

Acknowledgments

This dissertation could not see the light without the contributions of many people who supported me along the way. First and foremost, I would like to express my gratefulness to my advisor, Prof. Parimala Thulasiraman for her insights and guidance. During the many one-to-one discussions, Dr. Thulasiraman enriched my knowledge and opened my eyes to new approaches in the fields of Swarm Intelligence and High-Performance Computing. She has always cleared several obstacles and made this research very enjoyable. In fact, this dissertation would not have been possible without her supervision. I owe Dr. Thulasiraman much for her insightful remarks, understanding, encouragement, patience, friendship, and kindness. Besides, I would like to thank my committee members, Dr. Udaya Annakkage and Dr. Yang Wang, for their remarkable comments and valuable advice on this research. I am also thankful to the external examiner, Prof. Laurence T. Yang, Computer Science, St. Francis Xavier University, for dedicating considerable time and attention to my dissertation.

Many thanks are due to many outstanding professors and graduates in the department of Computer Science at UFM for their respect, knowledge, and integrity. Special thanks are also due Dr. Ruppa Thulasiram for supervising my first two papers. Lastly, this dissertation would not be possible without the endless support from my wife, Mrs. Nesreen Samhan.

Chapter 1

Introduction

The proliferation of ubiquitous systems over the last two decades has led to the emergence of "Big Data" era. In 2012, Gartner Inc. defined Big Data as "high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation." (De Mauro et al. [2015]). Based on this definition, Big Data presents challenges along three dimensions: volume (rapid increase in data size), velocity (real-time data changes) and variety (data generation from heterogeneous sources in various data types). These inherent challenges require new innovative approaches to extract meaningful, useful, and often vital information from the massive amounts of raw data.

There are two fundamental approaches to extract useful information from data: supervised learning (classification) and unsupervised learning (clustering). Standard classification algorithms (e.g., logistic regression, naive Bayes, support vector machines) rely on learning a classifier (mathematical function) from correctly identified observations (training data), to predict a predefined class label for an unseen observation. For instance, a dataset of emails labeled as spam or non-spam can be used for training a naive Bayes classifier to predict whether a new email is a spam or non-spam. Although classification algorithms have been successfully implemented in various domains, they suffer from the deficiency of training data and the high cost of hand-labeling.

Clustering approaches, on the other hand, are entirely unsupervised (i.e., no training data required). They aim at finding intrinsic structures in unlabeled data. The objective of the clustering methods is to partition a set of data instances into unknown number (k) of mutually exclusive clusters according to some optimality criterion. Typically, the instances (objects) within the same cluster should be as similar as possible, and they should be as dissimilar as possible from instances in other clusters. The similarity/dissimilarity of objects is often measured by a distance function. Generally, the goal of the clustering is to optimize intra-cluster similarity and inter-cluster dissimilarity simultaneously. In this thesis, my focus is on clustering.

The computational complexity of finding an optimal clustering solution is proven to be NP-hard (Welch [1982]). The number of feasible solutions grows exponentially with respect to the number of data instances to be clustered. As a result, many clustering approaches have been proposed. There are many exact algorithmic clustering approaches. For example, connectivity-based (hierarchical), centroid-based (partitioning or k-means), graph-based (Clique), distribution-based (expectation-maximization), density-based (DBSCAN and OPTICS) and spectral-based clustering. Typically, there is no single approach that is appropriate for all types of data, nor are all approaches suitable for all problems. Each clustering approach has its shortcomings concerning object heterogeneity, efficiency, simplicity, and scalability. For instance, k-means algorithm, the most commonly used technique due to its ease in implementation, suffers from convergence to a local optimum as the outcome is highly affected by the selection of initial partitions (Zhao and Karypis [2004]).

Exact algorithmic clustering approaches are not affordable for many real-world applications that require innovative approximation algorithms. Among them are meta-heuristics such as bio or nature inspired techniques (Glover and Kochenberger [2006]). Some examples of meta-heuristics include: simulated annealing, tabu search, genetic/evolutionary algorithms, variable neighborhood search, (adaptive) large neighborhood search and ant-based systems. According to (Glover and Kochenberger [2006]), meta-heuristics are "master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality". Unlike exact and approximate algorithms, meta-heuristics are always heuristic in nature. Consequently, they do not guarantee that an optimal solution will be found, even though with a large amount of time. As a result, meta-heuristics are developed specifically to find a solution that is good enough in a reasonable computational time.

There are two major components of any meta-heuristic algorithm: intensification (exploitation) and diversification (exploration). Diversification is the ability of the algorithm to generate diverse solutions to explore the search space globally, whereas intensification aims at focusing the algorithm to search in a local region giving that a current good solution exists in this region. To improve the convergence rate of a meta-heuristic algorithm, there should be a good balance between intensification and diversification. Finding this balance ensures that solutions will converge to the optimum, while diversification via randomization allows the algorithm to escape from local-optima and, at the same time, increases the diversity of solutions. A good combination of these two major components will usually ensure that global optimality is achievable (Blum and Roli [2003]).

There are three benefits of meta-heuristics: (i) they are often able to offer a better trade-off between solution quality and computational time; (ii) they can be adapted to match the requirements of most real-life optimization problems because meta-heuristics offer a high-level problem-independent algorithmic framework; (iii) meta-heuristics do not need a formulation for the optimization problem (i.e., (specify the problem in the form of constraints and objective functions).

One of the many meta-heuristic approaches used for solving the clustering problem is swarm intelligence (SI). SI is an artificial intelligence paradigm inspired by the behavior of real swarms or insect colonies such as ant colonies, bird flocking, animal herding, bacterial growth, or fish schooling. In SI, the organisms (agents) modeled in the system, work independently providing lots of parallelism. The agents, distributed within the environment, co-operate/co-ordinate through stigmergic or indirect communication reducing global communication (Navlakha and Bar-Joseph [2015]) and providing data locality. They self-organize when needed and work asynchronously within their local environments. These characteristics make SI techniques quite amenable to many real world applications such as community detection. One of the swarm intelligence techniques that has been studied in the literature to solve the clustering problem in community detection application is ant colony optimization (ACO) algorithm (Honghao et al. [2013]). However, the algorithm does not work well for large networks.

In this thesis, I consider another technique, ant brood clustering (ACA) (Deneubourg et al. [1991]; Lumer and Faieta [1994]) inspired by how real ants brood sort their nest. This technique is more suitable to the clustering problem than ACO. In (Liu and Liu [2016]), Liu provides an in-depth study of the algorithm providing its weakness and strength through many benchmark experimentations. However, there are some shortcomings to the original ACA mathematical model by Lumer and Faieta (LM model) for it to be applicable to real world applications: (i) user defined parameters have to be "experimentally" fine-tuned to reflect the application under study; (ii) the ant's radius of perception is assumed constant - narrowing an ant's visibility, consequently, converging to a local optimum; (iii) lack of communication between ants prevents the ants from dropping the object in the best location. As a result, ants perform redundant searches within their local neighborhood, until they find a location that satisfies the object dropping criteria which is computationally intensive. In this thesis, I present an Ant brood Clustering Algorithm with Adaptive Radius of perception (ACA-AR), a modified variant of ACA for clustering multi-dimensional data. ACA-AR uses multivariate kernel density estimation and sigmoid function to improve the estimation of ants' pick-up and drop-off probabilities. As a result of these modifications, ACA-AR gains many advantages over traditional and ACA clustering existing models. ACA-AR

- does not make any prior assumptions about the number or the shape of the clusters.

- converges to the exact number of clusters in the data since it balances the trade-off between maximizing inter-cluster dissimilarity and intra-cluster similarity.
- detects data outliers using adaptive radius strategy.
- avoids convergence to local-optima solutions.
- substantially improves the spatial separation of clusters on the grid, an essential requirement to retrieve the clusters.
- preserves the characteristics of nature-inspired algorithms, making it suitable for clustering data in dynamic domains (Wang et al. [2009]).

In many real-world applications, gathering unlabeled data is cheap and easy while extensive hand-labeling of data is costly and time-consuming. However, in many applications, it is possible to acquire small amount of prior knowledge (extra side information, small-size labeled data) that specifies whether the particular data instances are similar or dissimilar to cluster them. In such cases, neither applying supervised nor unsupervised clustering is feasible. Therefore, many traditional clustering algorithms have been extended to semi-supervised settings so they can take advantage of the prior knowledge to supervise or "guide" the clustering process. For example, in protein function prediction, some pairwise constraints can be identified experimentally by finding functional links between pairs of protein genome sequence data (Eisenberg et al. [2000]). Satisfying constraints in data clustering is significant to reflect the object similarity within the domain. In most cases, prior knowledge can be naturally expressed in the form of instance-level pairwise constraints. Clustering in the presence of limited supervisory knowledge encoded in the form of pairwise constraints is known as *semi-supervised* or *constrained clustering* (Wagstaff and Cardie [2000]; Basu et al. [2008]). Please note that I use both terms interchangeably. Constrained clustering has proven beneficial in many real-world applications, such as lane finding from GPS traces, noun phrase co-reference resolution (Wagstaff et al. [2001]), and personal identification from surveillance camera clips (Bar-Hillel et al. [2005]).

In this thesis, I extend ACA-AR to constrained (semi-supervised) clustering algorithm (CACA-AR) to take advantage of pairwise constraints to further improve the estimation of pick-up and drop-off probabilities. I experimentally validate ACA-AR with and without constraints on three benchmark data sets that present different clustering challenges. The results show that ACA-AR outperforms ACA (Lumer and Faieta [1994]), mean shift and k-means algorithms in terms of clustering accuracy, completeness, and homogeneity. Moreover, the results show that the accuracy of ACA-AR substantially improved when pairwise constraints are incorporated, especially when clustering high-dimensional datasets.

As a case study, I evaluate the application of CACA-AR to the tasks of aspect category identification and sentiment prediction in the domain of product reviews, central clustering tasks in aspect-based sentiment analysis (ABSA), by formulating both tasks as constrained clustering problems. The results illustrates CACA-AR effectiveness to real-world applications such as ABSA.

1.1 Contribution

My contribution to the thesis are as follows:

1. Develop an adaptive clustering algorithm, called *ACA with Adaptive Radius (ACA-AR)* using kernel density estimation, a non-parametric statistical model, to measure average dissimilarity of data objects in ants neighborhood.

2. Parallelize ACA-AR on mutli-core machines.

3. Develop a search-based semi-supervised constrained clustering algorithm (CACA-AR) that incorporates supervisory information to guide the clustering algorithm towards solutions where constraints are minimally violated.

4. Evaluate the accuracy of CACA-AR on benchmark datasets.

5. Provide a feasibility study on one real world application, aspect based sentiment analysis.

Chapter 2

Literature Review

This chapter is organized into four sections. Sections 2.1 and 2.2 introduce the problem and definition of constrained clustering. Section 2.3 reviews the traditional semi-supervised clustering algorithms. Section 2.4 introduces ant-based clustering and reviews the constrained clustering studies based on ant clustering algorithm.

2.1 Constrained Clustering

The scarcity of labeled data and the high cost of obtaining such data are significant obstacles in applying standard classification algorithms in many real world applications. However, such applications make available large amounts of unlabeled data as well as small quantity of supervisory information that can be naturally expressed in the form of sets of pairwise constraints. In protein function prediction, for example, pairwise constraints can be identified experimentally by finding functional bonds between protein genome sequence data (Eisenberg et al. [2000]). These relationships reflect the experts' perspective on object similarity in the domain as they indicate whether particular protein genome sequences (data objects) are similar or dissimilar to be grouped in the same or different clusters.

Constrained clustering, also known as semi-supervised clustering, aims at enhancing clustering outcomes by incorporating instance-level pairwise constraints in a clustering algorithm. Constraints help to specify whether two data instances can be clustered together. (Basu et al. [2004a]). Constraints can be either explicitly defined by a domain expert or extracted from small labeled datasets. In this thesis, I investigate the use of *instance-level pairwise constraints* to improve clustering quality of my proposed ant clustering algorithm with adaptive radius (ACA-AR).

2.1.1 Instance-Level Pairwise Constraints

According to (Wagstaff et al. [2001]), there are two types of instance-level pairwise constraints: either two data instances, A and B, can be declared to be in the same cluster, called *Must-linked*, $ML(A, B)$, or can be declared to be in different clusters, called *Cannot-linked*, $CL(A, B)$. Both types of instance-level constraints exhibit different properties. A set of ML-constraints, for instance, is symmetric, reflexive and transitive. The transitivity property allows expanding ML set by inferring more ML-constraints. For example, given data instances (A, B, C, D) and two ML-constraints $ML(A, B)$ and $ML(B, C)$ as shown in Figure 2.1, by the transitive closure of ML, we can induce the constraint $ML(A, C)$. Transitivity of ML-set can be generalized by assuming that data instances represent nodes in an undirected graph, and each $ML(A, B)$ represents an edge between data instances (A, B). ML-constraints can be identified as the connected components in the graph. Moreover, if there exist an edge (ML-constraint) between two nodes in two different connected components CC_i and CC_j, then we can infer that each node in CC_i is also connected by a ML-constraint with the nodes in CC_j and vice versa (Basu et al. [2008]).

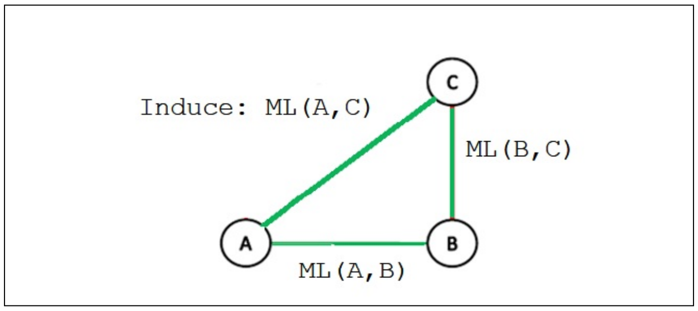


Figure 2.1: Graph Representation for ML Transitive Closure

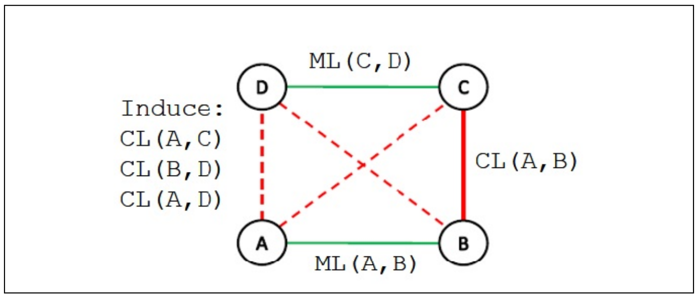


Figure 2.2: Graph Representation for CL Entailment

A set of CL-constraints, by contrast, is not transitive; the existence of $CL(A, B)$ and $CL(B, C)$ does not imply $CL(A, C)$. Nevertheless, CL set can be expanded using CL entailment property as illustrated in Figure 2.2. Given $ML(A, B)$, $ML(C, D)$ and $CL(B, C)$, the constraints $CL(A, C)$, $CL(B, D)$ and $CL(A, D)$ can be induced. Also, this property can be generalized by combining CL and ML constraints in one graph. Let CC_i and CC_j be two connected components in a ML graph. If there exist at least one CL-constraint between CC_i and CC_j, we can infer the existence of CL-constraints for all nodes in CC_i and CC_j (Basu et al. [2008]).

Formally, given a set of n data instances $\{x_i\}_{i=1}^n$, set of ML-constraints $ML = \{(x_i, x_j)\}$, and set of CL-constraints $CL = \{(x_i, x_j)\}$, the dataset $\{x_i\}_{i=1}^n$ can be represented as a graph, $G(V, E)$, such that $V = \{x_i\}_{i=1}^n$ and $E = ML$. As mentioned above, applying the transitive closure on ML set results in forming connected components in G. Moreover, $if\ \exists ML(x_i, x_j)\ s.t.\ a \in CC_i, b \in CC_j\ then\ \forall x_i \in CC_i, x_j \in CC_j \rightarrow ML(x_i, x_j)$ where CC_i, CC_j are two different connected components in G. Similarly, $if\ \exists CL(x_i, x_j)\ s.t.\ x_i \in CC_i,\ x_j \in CC_j \rightarrow CL(x_i, x_j)\ \forall x_i \in CC_i,\ \forall x_j \in CC_j$ (Basu et al. [2008]).

2.1.2 Constraint Extraction

Constraint extraction refers to the process of getting constraints either manually or automatically for a particular domain. In the manual methods, we ask a user, who is usually a domain expert, to determine whether a pair of data instances can be related by a must-link or a cannot-link constraint. However, providing the user with many data pairs makes this process tedious and hence error-prone. Therefore, the user is given only a small randomly selected subset of data instances, from which the must-link and cannot-link pairs are determined. This random selection of constraints may be ineffective because any clustering algorithm can trivially determine the relation of some selected pairs of data. As a result, different active learning methods have been proposed to identify the most informative pairs of data, such as (Basu et al. [2004a]).

Automatic methods, on the other hand, recognize constraints by finding which pairs of data instances are similar or dissimilar enough to be associated. Consequently, automated methods themselves are clustering algorithms with domain-dependent rules. These rules provide information that is not captured by data representation or similarity measures. They can thus be obtained by analyzing properties of data instances or by using external domain sources.

2.2 Formal Definition of Constrained Clustering

Given a set of n data instances $\{x_i\}_{i=1}^n$, where x_i is a real-valued vector of dimension d; a set of must-linked constraints $ML = \{(x_i, x_j)\}$ such that $i \neq j$, and a set of cannot-linked constraints $CL = \{(x_i, x_j)\}$ such that $i \neq j$. The problem of constrained clustering is to partition $\{x_i\}_{i=1}^n$ into k disjoint clusters $C_1, C_2, ..., C_k$ according to optimality criterion, such that the following criteria are satisfied for all $1 \leq i \leq k$

- $\bigcup_{i=1}^{k} C_i = \{x_i\}_{i=1}^n$.
- $\bigcap_{i=1}^{k} C_i = \Phi$
- $\forall (x_i, x_j) \in ML$; x_i, x_j are in the same cluster C_i; $i = 1, 2, ..., k$
- $\forall (x_i, x_j) \in CL$; x_i, x_j are in the different clusters C_i, C_j; $i, j = 1, 2, ..., k$ $i \neq j$
- intra-cluster similarity is maximized
- inter-cluster dissimilarity is maximized

The problem of constrained clustering is to find an optimal or near-optimal clustering solution C^* with respect to a fitness function (similarity function) such that the intra-cluster similarity is maximized, the inter-cluster dissimilarity is maximized, and the given constraints, must-linked and cannot-linked, are satisfied. Finding C^* is NP-hard problem. Regardless of whether the value of k is known or not (Davidson and Ravi [2005a]; Davidson and Ravi [2005b]), the number of feasible solutions grows exponentially with respect to the number of data instances n to be clustered.

2.3 Constrained Clustering Algorithms

While unsupervised clustering algorithms are common and diverse, semi-supervised clustering algorithms are limited and have a short history. According to Zhu et al. (Zhu [2005]), existing constrained clustering algorithms fall into two major approaches: search-based and similarity-adapting. The primary distinction between the two approaches relies on how constraints are utilized to guide the clustering algorithm.

2.3.1 Search-based Methods

In this approach, constraints are primarily used to bias an existing search-based clustering algorithm towards more appropriate data partitioning. Different methods have been proposed to achieve this end.

Pairwise-constrained k means

One of the earliest methods, such as the pairwise-constrained k-means (PCK-means (Wagstaff et al. [2001])) strictly enforces constraints during the assignment of data instances to the cluster centroids in the k-means algorithm. However, strict enforcement of both ML and CL constraints is proven to be an NP-complete problem (Basu et al. [2008]). In another method (MPCK-Means (Bilenko et al. [2004]), for example, constraints are employed to select the initial cluster centroids in the k-means algorithm. MPCK-means starts with finding the k-largest connected components in constraint graph to initialize cluster centroids. Constraint graph is constructed such that each data instance represents a node where those nodes participating in ML constraints are connected. Each data instance is then assigned to the cluster centroid that minimizes both the similarity distance (e.g., Euclidean) and the constraint violation (Bilenko et al. [2004]). To achieve this goal, MPCK-means adds two penalty weights that measure how often ML or CL constraints are violated to the k-means fitness function.

Probabilistic constrained clustering

The probabilistic constrained clustering model proposed by(Basu et al. [2004b]). It relies on using hidden Markov random fields (HMRF) to incorporate constraints in the k-means algorithm. The HMRF k-means generalizes PCK-means by combining constraints and Euclidean distance learning. It also allows the use of a broad range of clustering distortion measures. HMRF k-means aims at minimizing an objective function that is derived from the posterior energy of the HMRF model. The objective function involves the Euclidean distance between a data instance and a cluster centroid, a weighted penalty factor for violating constraints, and a normalization factor.

Spectral Constrained Clustering

Constraints have been incorporated in spectral clustering algorithms such as normalized cuts (NC) (Shi and Malik [2000]) giving rise to constrained normalized cuts (CNC) clustering (Basu et al. [2004b]).

Normalized cuts (NC) is a spectral clustering algorithm based on converting the clustering problem into a weighed graph partitioning problem. The graph is created such that data instances represent nodes and the degree of similarity between data instances represent edge weights. Different strategies have been proposed to connect nodes. One strategy is to connect two nodes if the similarity degree, as given by a similarity function, between the corresponding data instances exceeds a user-defined threshold ϵ. Another strategy is to connect each node to its k-nearest neighbors.

Formally, let $G = (V, E)$ be undirected, weighted graph with weight adjacency matrix W. Given a set A such that $A \subset V$, and its complement $\overline{A} = V \backslash A$. A is connected if for any pair of vertices $(v_i, v_j) \in A$, there exists a path that connects v_i and v_j, and all intermediate vertices lies in the path are in A. A is a *connected component* if and only if A is connected and there are no connections between A and $\overline{A}$. Spectral clustering techniques aim at finding graph cuts $\{A_1, A_2, ..., A_k\}$ (i.e., finding cuts in the form of connected components in the graph) such that $A_i \cap A_j = \phi$, $A_1 \cap, ..., \cap A_k = V$, the intra-partition similarity is maximized, and inter-partition similarity is minimized. Several objective functions have been proposed to encode such optimization. The most common ones are the ratio cut $RatioCut(A_1, A_2, ..., A_k)$ (Wei and Cheng [1991]) and the normalized cut $Ncut(A_1, A_2, ..., A_k)$ (Shi and Malik [2000]). In the former, the size of the cut A_i is measured by the number of vertices $|A_i|$, while the latter measures the size of the cut by the weights of the edges $vol(A_i) = \sum_{j \in A_i} w_{ij}$. According to (Shi and Malik [2000]), the weight of a cut $w(A_i, A_j)$ is defined as:

$$w(A_i, A_j) = \sum_{i \in A_i, j \in A_j} w_{i,j}$$

For the cuts $A_1, A_2, ..., A_k$ in G, the inter-cut similarity can be measured by:

$$Ncut(A_1, A_2, ..., A_k) = \frac{1}{2} \sum_{i=1}^{k} \left(\frac{w(A_i, \overline{A_i})}{vol(A_i)} \right)$$

However, finding a solution $\{A_1, A_2, ..., A_k\}$ that minimizes $Ncut$ is NP-hard problem (Shi and Malik [2000]). Nevertheless, a solution can be found in polynomial time using spectral graph techniques which make use of the eigenvectors of the graph Laplacian matrix. Such eigenvectors can be perceived as a low-dimensionality representation of the graph. Therefore, they can be used to cluster data instances in fewer dimensions. To achieve this, minimizing $Ncut$ is formulated as a standard matrix trace minimization problem ([Von Luxburg, 2007]). Given a partition of V into cuts $\{A_1, A_2, ..., A_k\}$, assume that H is $n \times k$ matrix that indicates the membership of data instances n to the cuts k. Each column j of H is encoded as follows:

$$h_{ij} = \begin{cases} \frac{1}{\sqrt{vol(A_j)}} & if v_i \in A_j \\ 0 & \text{otherwise} \end{cases}$$

Let $D_{n \times n}$ be a diagonal matrix such that $d_{ii} = desgree(v_i) = \sum_{j=1}^{n} w_{ij}$, and let $L = D - W$ be the unnormalized Laplacian matrix of graph G. Given the fact that $H^T H = I$, $h_i^T D h_i = 1$, and $h_i^T L h_i = cut(A_i, \overline{A_i})/vol(A_i)$, the minimization problem of $Ncut$ can be written as:

$$\min_{(A_1,A_2,\dots,A_k)} Tr(H^T LH) \; subject \; to \; H^T DH = I$$

This trace minimization problem can be written in the standard form (as given below) by relaxing H to take arbitrary real values in $\mathbf{R}^n$ and substitute $Y = D^{\frac{1}{2}}H$ (Von Luxburg [2007]).

$$\min_{Y\in\mathbf{R}^{n\times k}} Tr(Y^T \big(D^{\frac{-1}{2}} LD^{\frac{-1}{2}}\big)Y) \; s.t. \; YY^T = I$$

The standard trace minimization can be solved by choosing Y as the matrix which contains the first k eigenvectors of $D^{\frac{-1}{2}}LD^{\frac{-1}{2}}$ as columns (Von Luxburg [2007]). The final data clustering is achieved by recovering $H = D^{-1/2}Y$. It has been noticed that the solution H, consists of the first k generalized eigenvectors of $Lu = \lambda Du$. The final clusters are achieved by clustering the rows of H using k-means. This results in the normalized spectral clustering algorithm (Shi and Malik [2000]).

Normalized cut algorithm is extended to the constrained normalized cut (CNC) by Wang et al. ([Wang and Davidson, 2010]; Wang et al. [2014]). CNC is based on modifying the objective function of NC algorithm so that it optimizes graph cuts and satisfies constraints simultaneously. In more detail, in CNC, ML and CL constraints are encoded in the form of a symmetric matrix $Q_{n\times n}$ as follows:

$$Q_{ij} = Q_{ji} \begin{cases} 1 & \text{if } ML(i,j) \\ \text{-1} & \text{if } CL(i,j) \\ 0 & \text{otherwise} \end{cases}$$

Let $u \in \{1, -1\}^n$ be a cluster indicator vector, such that a data instance i belongs to the cluster if $u_i = +1$, and does not belong to the cluster if $u_i = -1$. Then, the value of $u^TQu = \sum_{i=1}^{n}\sum_{j=1}^{n} u_i u_j Q_{ij}$ indicates how well the constraints in Q are satisfied in the cluster indicator u. This measure increases by 1 each time a constraint is satisfied because $Q_{ij} = 1$ and data instances $i,\ j$ have the same sign in u. Conversely, the measure decreases by 1 each time a constraint is violated since $Q_{ij} = -1$ and $i,\ j$ have different signs in u. This encoding scheme is extended by relaxing both Q and u to take arbitrary real numbers that reflect the degree/strength of constraint relationship. Q_{ij} is positive if (i, j) is in the same cluster, and Q_{ij} is negative if (i, j) is in different clusters. The larger the value of u^TQu, the better the cluster assignment respects the given constraints in Q. Similar to the normalized cut, Q is normalized by replacing u with $D^{-1/2v}$ so the lower bound becomes $v^T\overline{Q}v$ where $\overline{Q}$ is the normalized constraint matrix $\overline{Q} = D^{-1/2}QD^{-1/2}$.

Since satisfying all given constraints is intractable, the authors set a lower bound $u^TQu \geq \alpha$, where α is a constant threshold number $\alpha \in \mathbb{R}$. This lower bound is augmented to the objective function of normalized cut. Given a normalized graph Laplacian $\bar{L}$, a normalized constraint matrix $\overline{Q}$, and a threshold α, the objective function of CNC is:

$$\min_{v\in\mathbb{R}} v^T\bar{L}v \; \textit{subject to} \; v^T\bar{Q}v \geq \alpha, v^Tv = vol, v \neq D^{1/2}\mathbf{1}$$

In this optimization, $v^T\bar{L}v$ is the cost of the cut, $v^T\bar{Q}v \geq \alpha$ is a lower bound on constraint satisfaction, $v^Tv = vol$ is a normalization of V, and $v \neq D^{1/2}\mathbf{1}$ eliminates the trivial solution $D^{1/2}\mathbf{1}$. The authors follow the Karush-Kuhn-Tucker theorem (Kuhn [1982]) to solve this optimization problem with respect to the necessary conditions. This theorem is beyond the scope of my thesis.

2.3.2 Similarity-Adapting Methods

Similarity-adapting methods rely on modifying similarity measure in a given clustering algorithm so that the available constraints are easily satisfied.

In (Klein et al. [2002]), the authors modify similarity values computed by Euclidean distance to incorporate pairwise constraints called complete link agglomerative (CLA) clustering. CLA is an iterative algorithm that initially considers each data instance is a cluster. If two data instances are linked by a must-link constraint, their distance is set to zero in the similarity matrix. Similarly, if two data instances are linked by a cannot-link constraint, their distance is set to a maximum threshold distance. This adjusting of data instance similarity associated with pairwise constraints is called imposing constraints. Two similar clusters are merged in each iteration until one cluster is left. The similarity between two clusters is determined by the maximum distance between their corresponding data instances.The outcome of CLA is a hierarchy of clusters that is known as dendrogram. The dendrogram can be cut at the appropriate level to retrieve the desired number of clusters.

Another proposed similarity adjustment (Klein et al. [2002]) is to propagate constraints to the neighboring data instances. If x_i, x_j are two data instances that are very close to each other, then all data instances which are close to x_i are also close to x_j. Likewise, if x_i, x_j are far apart, then data instances which are close to x_i are also far from x_j. However, the direct imposing of must-link constraints in the similarity matrix results in violating the triangle inequality and the shortest path properties between data instances. To maintain these properties, authors apply all-pairs-shortest-paths algorithm on the imposed matrix to create a metric matrix. As for cannot-link constraints, authors state that finding a clustering that satisfies them is NP-hard. However, they argue that such constraints can be imposed and propagated implicitly by choosing a similarity-based clustering algorithm such as the CLA algorithm.

Constrained spectral clustering

Another significant similarity adapting work is proposed by Kamvar et al. (Kamvar et al. [2003]). In their work, the constraints are imposed in spectral clustering by combining data similarities with pairwise constraints to produce a Markov transition process between data instances. This Markov transition is achieved by turning the similarity matrix A into a normalized Markov transition process N. The eigenvectors of $N_{n \times k}$ are then used for detecting data blocks which correspond to clusters by projecting data instances into $\mathbf{R}^k$. To construct the matrix N from A, the following operation is applied:

$$N = \frac{1}{d_{max}}\big(A + d_{max}I - D\big)$$

where D is the diagonal matrix.

Semi-supervised constrained clustering

Cohn et al. (Cohn et al. [2003]) proposed a semi-supervised clustering based on user feedback. The basic idea is to cluster data instances using an unsupervised clustering algorithm. The user can criticize the clustering outcome by setting some constraints on the contents of clusters. These constraints are used for re-clustering data by changing the similarity distance metric of the clustering algorithm. This process continues until the user is satisfied with the results. In more detail, the authors present a prototype-based clustering derived from naive Bayes model of document generation. In their model, each document is represented as a “bag-of-words” that is generated from a multinomial distribution θ. The probability of a document d is given by:

$$P(d) = \prod_{w_i \in Vocabulary} p(w_i|\theta)^{N(w_i,d)}$$

where $p(w_i|\theta)$ is the probability that term w_i is generated, and $N(w_i, d)$ is the frequency of term w_i occurrence in d. Each document d is an estimate of a multinomial distribution θ_d, and each cluster π of documents is an estimate θ_π . As for the clustering, the authors assume that each document drawn from one distribution $\theta_{\pi_1}, \theta_{\pi_2}, ..., \theta_{\pi_k}$ correspond to the unknown cluster distributions $\pi_1, \pi_2, ..., \pi_k$.

$$P(d) = \sum_i P(\pi_i)P(d|\pi_i) = \sum_i P(\pi_i) \prod_{w_j \in Vocabulary} P(w_j|\theta_{\pi_1})^{N(w_j,d)}$$

The goal is to estimate the values of $P(\pi_i)$ and θ_{π_i} which in turn can be used to estimate class membership by Bayes rule:

$$P(\pi_i|d) = \frac{P(d|\pi_i)P(\pi_i)}{P(d)}$$

To implement pairwise document constraints, authors augment the standard KL-divergence $D_{KL}(\theta_{d_1}, \theta_{d_2})$ with a weighting function as follows:

$$D\prime_{KL}(\theta_{d_1}, \theta_{d_2}) = \sum_{w_j \in Vocabulary} \upsilon_j P(w_i|\theta_{d_1}) log \frac{P(w_j|\theta_{d_2})}{P(w_j|\theta_{d_1})}$$

$P(w_j|\theta_{d_2})$ indicates the importance of w_j for distinguishing d_1 and d_2. Given constraint that d_1 and d_2 must be in different clusters, the authors adjust the metric by:

$$\frac{\partial D\prime_{KLM}(d_1, d_2)}{\partial \upsilon_j} = |x_1| P(w_j|\theta_{d_1}) log\left(\frac{p(w_j|\theta_{d_1,d_2})}{p(w_j|\theta_{d_1})}\right) + |x_2| P(w_j|\theta_{d_2}) log\left(\frac{p(w_j|\theta_{d_1,d_2})}{p(w_j|\theta_{d_2})}\right)$$

The distance between d_1 and d_2 can be increased by hill-climbing over the υ. These υ's are then incorporated into the E-step of the clustering algorithms as weights attached to the term frequencies.

$$P(d|\pi_i) = \prod_{w_j \in vocabulary} P(d|\theta_\pi)^{\upsilon_j} N(w_j, d)$$

Learnable similarity measures

Other significant similarity-adapting works are presented in (Xing et al. [2003]; Bilenko and Mooney [2003]). Given a set of similar or dissimilar data instances, the work in (Xing et al. [2003]) relies on learning Mahalanobis distances adjusted by convex optimization to perform clustering that respects similarity/dissimilarity relationships. A distance metric in the form:

$$d(x, y) = d_A(x, y) = \big||x - y\big||_A = \sqrt{(x - y)^T A(x - y)}$$

To make d a metric that is non-negative and satisfies the triangle inequality, A has to be positive semi-definite ($A \succeq 0$). A parameterizes a family of Mahalanobis distances over $\mathbf{R}^n$. If $A = I$, we get the Euclidean distance. Learning such metric is also equivalent to finding re-scaling that replaces x with $A^{1/2}x$and applying the Euclidean distance to the re-scaled data. This can be formulated as constrained optimization:

$$\begin{aligned} \underset{A}{\text{minimize}} \quad & \sum_{(\mathrm{x_i},\mathrm{x_j})\in \mathrm{S}} (\big||x_i - x_j\big||_A^2) \\ \text{subject to} \quad & \sum_{(x_i,x_j)\in D} (\big||x_i - x_j\big||_A^2) \geq 1 \\ & A \succeq 0. \end{aligned}$$

The authors consider two cases for A. The first case is the $diagonal A$ where we want to learn $A = diag(A_{11}, A_{22}, ..., A_{nn})$. By using Newton-Raphson method, $g(A)$ is defined as:

$$g(A) = \sum_{(x_i,x_j)\in S} (\big||x_i - x_j\big||_A^2) - \log \left(\sum_{(x_i,x_j)\in D} (\big||x_i - x_j\big||_A^2) \right)$$

Optimizing g $s.t.$ $A \succeq 0$ can be solved efficiently using Newton-Raphson method. The second case is to learn the $full$ A. Using Newton-Raphson is computationally expensive ($\mathcal{O}(n^6)$). Therefore, authors use gradient decent and the idea of iterative projections.

2.4 Constrained Clustering based on Swarm Intelligence

Inspiration from nature has driven many creative solutions to challenging real-life problems. Clustering, in its purest form, is an optimization problem. One of the many approaches used for solving clustering problem is Swarm Intelligence (SI). SI is an artificial intelligence paradigm that is mainly inspired by the behavior of real swarms or insect colonies. SI depends on the collective action of decentralized, self-organized agents. Although these agents have no direct communication or centralized control, the indirect local interactions between such agents result in the emergence of intelligent global behavior that is unknown to the individual agents. Examples of natural systems of SI include ant colonies, bird flocking, animal herding, bacterial growth, and fish schooling (Kennedy [2011]). The most popular swarm intelligent algorithms used for data clustering are ant colony optimization, particle swarm optimization and flocks of agent based-clustering.

In this thesis, I consider another technique, ant brood clustering (ACA) (Deneubourg et al. [1991]; Lumer and Faieta) inspired by how real ants brood sort their nest. This technique is more suitable to the clustering problem than ant colony optimization. In (Liu and Liu [2016]), Liu provides an in-depth study of the algorithm providing its weakness and strength through many benchmark experimentations.

Most of the work on ant clustering is based on the LM model by Lumer and Faieta (Lumer and Faieta [1994]) described in section (2.4.1). In section 2.4.2, I review previous constrained clustering works that are based on ACA. In Chapter 3.1, I highlight the shortcomings of using LF model for data clustering and present three major enhancements to alleviate its shortcomings. In section 3.3, I extend the enhanced model to a constrained clustering model by incorporating constraints.

2.4.1 Ant Clustering Algorithm (LF Model)

Ant brood clustering was inspired from the observation that some species of ants have the ability to sort large corpses and eggs into clusters. This behavior was modeled by Deneubourg et al. (Deneubourg et al. [1991]) to enable real-world robots perform certain clustering tasks. In the basic model, the data objects are scattered on a two-dimensional grid and the ants randomly move withing this grid. Each ant decides to either *pick up* a free object or *drop off* a loaded object depending on the average similarity of the object with its neighboring objects.

The basic model is enhanced by (Lumer and Faieta [1994]) to work with multidimensional data that are comparable according to a measure of similarity (LF Model). The average similarity is estimated using a local average function as shown in equation (2.1). In this function, $d(o_i, o_j)$ is a similarity function that measures the distance between the object o_i and a neighboring object o_j (e.g. Euclidean distance) scaled by a constant $\alpha \in [0, 1]$. r is known as the radius of perception of the ant, and $(2r)^2$ is the area of ant's neighborhood, which is a normalization factor that represents the square area surrounding the object o_i. As shown in the pickup equation (2.2), the more dissimilar objects there are in the ant neighborhood, the more likely the objects will be picked up. Conversely, the more similar objects there are in the ant neighborhood, the more likely the objects will be dropped off, as given in equation (2.3). k_p and k_d are two parameters for adjusting pickup and drop-off probabilities, respectively. After several iterations, data clusters emerge from the collective and collaborative activities of the ants.

The LF model is considered as a standard ant clustering algorithm in mining data. In 2002, Handl and Meyer implemented ACA for classifying on-line documents based on their cosine similarity (Handl and Meyer [2002]). Wu et al. (Wu and Shi [2001]) combined ACA with k-means algorithm to achieve more accurate clustering results.

$$f(o_i) = \max\left(\frac{1}{(2r)^2}\sum_{o_j \in N(o_i)} (1 - \frac{d(o_i, o_j)}{\alpha})\ ,0\right) \tag{2.1}$$

$$P_{pick}(o_i) = \left(\frac{k_p}{k_p + f(o_i)}\right)^2 \tag{2.2}$$

$$P_{drop}(o_i) = \begin{cases} 2f(o_i) & if\ f(o_i) < k_d \\ 1 & \text{otherwise} \end{cases} \tag{2.3}$$

2.4.2 Constrained Clustering based on ACA

While there is a wide and diverse literature on traditional constrained clustering algorithms, there is few works on clustering based on ant brood constrained algorithm. To the best of my knowledge, only two works have been proposed, (Yang et al. [2012]) and (Xu et al. [2011]). In this section, I explain both works in detail, and highlight their drawbacks.

In (Yang et al. [2012]), the authors propose a novel consensus constraint-based clustering algorithm that incorporates pairwise (CL and ML) constraints in multi-ant colonies. Clustering ensemble has proven to improve the quality and robustness of clustering by combining multiple clustering solutions into a single solution. The authors propose two problems: (i) how to incorporate pairwise constraints in each ant colony; (ii) how to compute a new similarity matrix by incorporating the provided constraints. The proposed model is similar to the LF model. Data instances are distributed randomly on a two-dimensional grid where each ant is initially assigned at a random data instance. While ants are moving on the grid, they are either picking up or dropping off data based on certain probabilities. Unlike LF models, however, the model allows ants to move with varying speed. The probability of picking up or dropping off depends on the similarity of the current data instance with its neighboring data instances. When an ant at position r finds an object o_i at time t, the average local density $f(o_i)$ of objects that are similar to o_i is computed by equation 2.4:

$$f(o_i) = \max\left(0, \frac{1}{s^2} \sum_{o_j \in Neigh(o_i)} [1 - \frac{d(o_i, o_j)}{\alpha(1 + ((v-1)/v_max))}]\right) \tag{2.4}$$

$Neigh(o_i)$ refers to $(s \times s)$ square surrounding area around o_i. $d(o_i, o_j)$ is the similarity distance between o_i and o_j. α is a factor that defines the scale of similarity between objects. v is a parameter to control the speed of ants, and v_{max} denotes the maximum ant speed. Three different cases are considered for the speed of ants: ants move randomly at the same speed (i.e., v is constant for all ants), the speed of each ant is randomly sampled from $[1, v_{max}]$, and ants start from maximum speed v_{max} then decreases randomly to cool down. To compute picking up and dropping off probabilities, they use the standard Sigmoid function $f(x) = \frac{1}{1+e^{-x}}$ using $f(o_i)$ with $f(x)$ as a parameter as shown in equations (2.5;2.6):

$$P_{drop(o_i)} = sigmoid(f(o_i)) \tag{2.5}$$

$$P_{pick(o_i)} = 1 - sigmoid(f(o_i)) \tag{2.6}$$

As the average similarity decreases, the probability of picking up increases while the probability of dropping off decreases.

In (Yang et al. [2012]), the authors also incorporate pairwise constraints to guide the clustering process towards an accurate partition (search-based). To make use of ML constraints in dropping off, the authors count the number of ML constraints that relate o_i with every neighboring object o_j. The ant drops off o_i if $P_{drop(o_i)}$ is greater than a random number $r \in [0, 1]$ or if the number of ML constraints is greater than a given constant c_1. By contrast, the ant picks up the object if $P_{pick(o_i)}$ is greater than a random number $r \in [0, 1]$, or the number of CL constraints that relates o_i with every neighboring object o_j is less than a given constant c_2.

The proposed ensemble clustering in (Yang et al. [2012]) involves three components: constant-moving ants, random-moving ants and randomly-decreasing moving ants. Let O be the set of objects $o_1, o_2, ..., o_n$. Each clustering component q produces a vector $\lambda^{(q)} \in N_n$ that indicates clustering label for each object. Applying r clustering components result in r different labeling vectors. These label vectors can be mapped into a binary membership matrix $H_{n \times kr} \in \{0, 1\}$ such that columns represent cluster membership and rows represent objects. For any column j in H, all objects that belong to a particular cluster are assigned one in their corresponding entries. The membership matrix is an adjacency matrix which in turn can be transformed into a similarity matrix as follows:

$$S = \frac{1}{r} H H^T$$

After computing the similarity matrix S, ML and CL constraints are incorporated as follows: if $ML(o_i, o_j)$ then $S_{ij} = 1$ and if $CL(o_i, o_j)$ then $S_{ij} = 0$. However, the authors do not explain how data instances are assigned with cluster labels.

There are many drawbacks in Yang et al. work. (i) varying speed of the ants does not improve the spatial separation among clusters. Thus, the algorithm probably converges to a local optimum; (ii) there is no integration between density function and constraints. This means that ants can drop an object within a neighborhood that has CL constraints with object, and can pick up an object from a neighborhood that has ML constraints with the object; (iii) no method is given to explain how to determine the values of constants c_1, c_2.

In ([Xu et al., 2011]), the authors introduce the Random Walk Ant Clustering (RWAC) algorithm. In RWAC, each ant represents a data point, and can randomly walk on the grid until it finds a place to sleep. Each ant perceives the fitness of the neighborhood to decide whether to sleep or continue moving. The behavior of finding a sleep place is simple. While ant moves, it looks for a safer place. The safety of the place is measured by the number of similar ants in the surrounding area of the ant. The authors improve RWAC convergence using a heuristic walk mechanism. In addition, they introduce constrained ant clustering (CAC) as a semi-supervised clustering algorithm. CAC uses heuristics to determine the direction of ant movement (left, right, up, down). Two heuristics are proposed: max-number direction walking (MNDW) and adaptive direction walking (ADW). The former simply counts k-nearest neighbors in each direction, whereas the latter involves the grid distance and the similarity in attribute space between the current ant and its neighbors are taken into account. RWAC also suffers from converging to local-optimum. The model forms larger number of small dense clusters than what originally exists in data. Also, it cannot separate the overlapped clusters.

Chapter 3

Ant Clustering Algorithm with Adaptive Radius (ACA-AR)

In section 2.4.1, the LF model for ant-based clustering was presented. In this chapter, I highlight its shortcomings, and present a modified version of the algorithm, called Ant Clustering Algorithm with Adaptive Radius (ACA-AR), that alleviates the shortcomings (sections (3.1,3.2)). I extend ACA-AR into a constrained clustering algorithm (CACA-AR) in section 3.3. In section 3.4, I explain the cluster retrieval process. Finally, I validate the ACA-AR with and without constraints on benchmark datasets in section 3.5.

3.1 Shortcomings of LF Model

The LF model suffers from the following shortcomings:

(i) the three initial parameters (α, k_p, k_d) in equations (2.1), (2.2) and (2.3) have to be "experimentally" fine-tuned. Since ant behavior is collective, a slight change in any one of the three parameters results in significant change in the outcome. Therefore, LF model is highly sensitive to the values of initial parameters which makes tuning these parameters problematic.

(ii) the model assumes that the ant's radius of perception is constant (i.e., fixed integer value). This assumption results in narrowing an ant's visibility; consequently, the ants spend tirelessly wandering on the grid and sometimes do not converge.That is, the ants move on the grid without picking up data objects. For instance, using ants with small constant radius (e.g., $r = 1$) for few hundreds of ant steps (iterations) results in forming a higher number of small dense clusters than what originally exist in the original data. On the contrary, assigning ant radius to higher values results in forming a lower number of small, sparse clusters than what originally exists in the data. This behavior is illustrated in Figure 3.1

(iii) the the model considers ants with a certain number of steps. In existing ACA models, the algorithm terminates when the ants reach a fixed number of random iterations (ant steps). In my modified algorithm, ACA-AR, I assume that ants are tireless, and they continue moving on the grid, picking up and dropping-off data items until the grid becomes stable.

As a result of aforementioned shortcomings, unloaded ants spend lots of time randomly moving on the grid without being able to pick up data items. Similarly, loaded ants spend lots of time to find a location that satisfies the object dropping criteria. This makes ACA computationally intensive and sometime redundant.

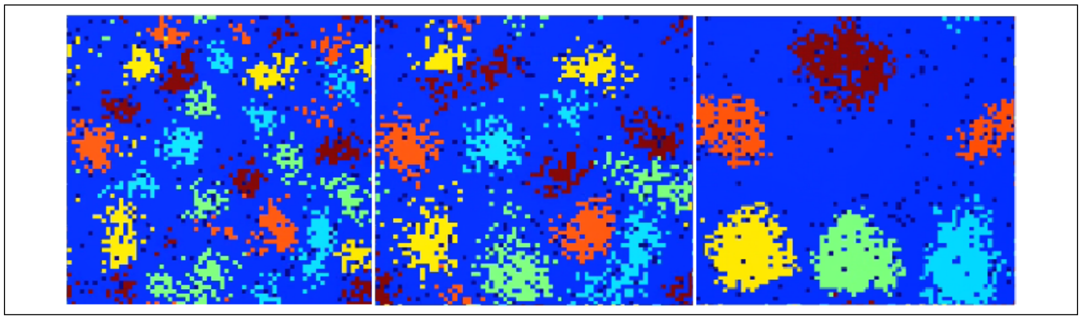


Figure 3.1: Color-coded clustering solutions for a mixture of Gaussian. Five classes of 1500x20 real-valued vectors, using 150 ants on a 25x25 grid. The left figure is the solution with ant radius 1, the middle figure is the solution obtained with ant radius 6, the right figure is a near-optimal solution obtained by ACA-AR by varying radius from 1 to 7

3.2 Enhancements to Ant Brood Clustering

This section presents three major enhancements to the LF model, constituting, ant clustering algorithm with adaptive radius (ACA-AR). The goal of these enhancements is to improve performance, stability, convergence, speed and robustness of ACA making ACA-AR applicable to real-world applications.

3.2.1 Applying Kernel Density Estimation to the Ant Neighborhood

As I previously described in section 2.4.1, equation (2.1) estimates the average similarity of ant's neighborhood. In ACA-AR, I use Kernel Density Estimation (KDE), a non-parametric statistical model that estimates the probability density function of a random variable. KDE is also known as the Parzen-Rosenblatt window method. For the univariate KDE, given a kernel function K and bandwidth parameter h, KDE estimates the probability density of a particular object x_i with respect to each neighboring object $x_j \in N(x_i)$ as shown in equation (3.1). For an object x_i at location (x, y), $N(x_i)$ is the set of objects within the area $[x \pm r, y \pm r]$. Aggregating these probability densities gives an overall picture of the underlying structure of the data and its density function.

$$KDE(x_i) = \frac{1}{nh^d} \sum_{x_j \in N(x_i)} K\left(\frac{\left\|x_i - x_j\right\|^2}{h}\right) \tag{3.1}$$

The univariate KDE can be generalized to the multivariate KDE as shown in equation 3.2. In this equation, K is a multivariate Kernel function with a bandwidth matrix H, $x \in \{x_i \in R^d\}_{i=1}^n$. The bandwidth matrix H can be selected using plug-in method or smoothed cross validation (Duong and Hazelton [2005]).

$$KDE_H(x_i) = \frac{1}{n} \sum_{x_j \in N(x_i)} \frac{1}{|H|} K(H^{-1}(x_i - x_j)) \tag{3.2}$$

For instance, substituting the Gaussian kernel in (3.1) gives equation (3.3) where the standard deviation parameter σ works as the bandwidth parameter, h. Similarly, substituting the Gaussian kernel in (3.2) gives equation (3.4) where the covariance matrix is the bandwidth matrix, H.

$$KDE(x_i) = \frac{1}{nh^d} \sum_{x_j \in N(x_i)} e^{\frac{-\left\|x_i - x_j\right\|^2}{2\sigma^2}} \tag{3.3}$$

$$K_H(x) = (2\pi)^{-d/2} |H|^{-1/2} \exp(-\frac{1}{2} x^T H^{-1} x) \tag{3.4}$$

Using KDE in ACA is more convenient than the average similarity function (equation (2.1)) not only because it eliminates the need for the initial parameters, but also it enables the use of different kernel types (e.g. Gaussian, Linear, Polynomial, etc.) for different clustering purposes.

If an ant needs to decide to pick up or drop off object x_i, it first computes $KDE(x_i)$ with respect to the set of objects in its neighborhood $N(x_i)$. $KDE \in [0, 1]$, with 1 being the maximum density of similarity. As shown in Figure 3.2, the more dissimilar objects there are in ant's neighborhood, the lower the value of KDE and hence the greater the probability of picking up. Conversely, the more similar objects there are in the ant neighborhood, the higher the KDE value and thus the greater the probability of dropping off. The picking and dropping values are bounded in $[0, 1]$ using the Sigmoid function as shown in equations 3.5 and 3.6 receptively:

$$Pr_{pickup}(x_i) = 1 - Pr_{drop}(x_i) \tag{3.5}$$

$$Pr_{drop}(x_i) = \frac{1 - \exp(-cKDE(x_i))}{1 + \exp(-cKDE(x_i)} \tag{3.6}$$

In equations 3.5 and 3.6, c is a constant that controls the convergence speed of the algorithm convergence when increased. As shown in the Figure, as c increases, the drop-off curve rapidly converges to 1, while the pick-up curve converges to 0 (i.e., drop and pickup curves become asymptotic to 1 and 0 receptively, for lower density values). As a result, the higher the values of c, the greater the probability of drop-off and the lower the probability of pick-up objects.

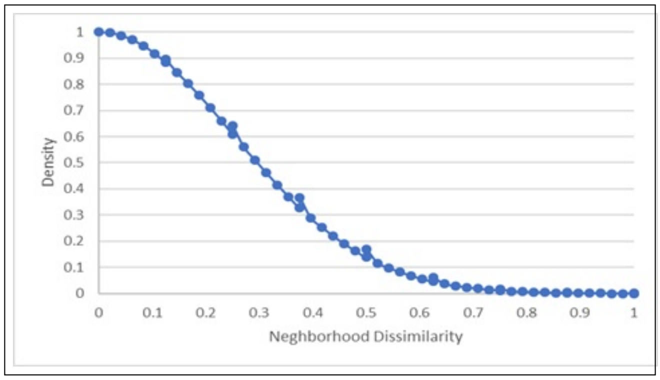


Figure 3.2: Dissimilarity Estimation of Ant Neighborhood Using KDE with Euclidean distance

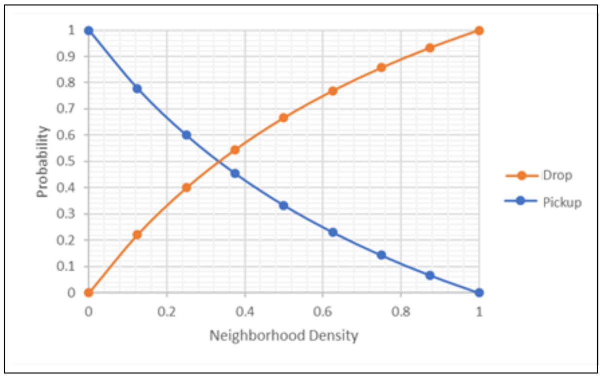


Figure 3.3: Pick-up and Drop-off Probabilities using Gaussian KDE for Neighborhood Density

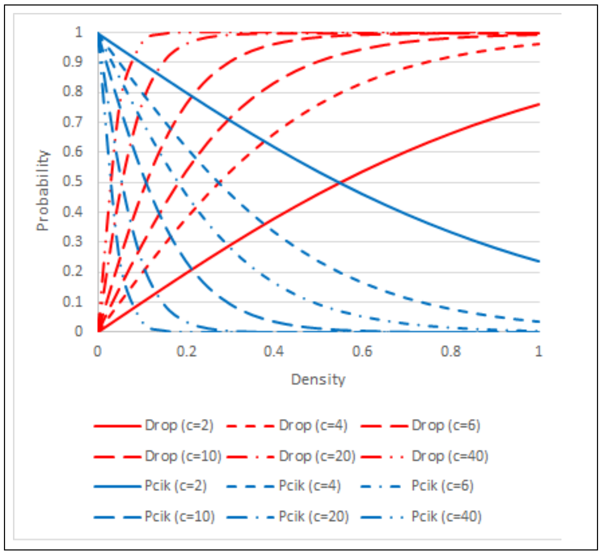


Figure 3.4: Pick-up and Drop-off Probabilities for different values of c

3.2.2 Adaptive Radius-based Ants

The radius of perception (r) determines the area of ant neighborhood; wherein, the ant can explore the nearby objects (i.e., ant's visibility) to decide its appropriate action. The ant's action can be either picking up a free object available at its current position on the grid, dropping off a loaded object if the ant position is empty, or moving to another random grid position. Such area is defined by a neighborhood function that can be as simple as a radius-based area or as complex as a copula. Regardless of the complexity of this neighborhood function, however, the outcome and convergence rate of ACA is highly sensitive to the value of ant radius of perception because it significantly affects KDE. For instance, setting the radius to a low value (e.g., $r = 1$ or $r = 2$) results in forming a greater number of dense clusters than what originally exists in the data. In contrast, high radius values (e.g., $r \geq \frac{1}{4}\sqrt{grid\ area}$)

yield a less number of sparse clusters than what originally exists in the data.

To solve this problem, I incorporate an ant with adaptive radius. That is, each ant can increase or decrease its radius of perception as it becomes unable to perform pickups. In more detail, all ants start wandering on the grid with $r = 1$ until they become unable to pick up objects. At this point, the outcome will be a local-minima solution (a large number of small dense clusters). This convergence occurs because most ants become moving without performing pickups. To stimulate ants to pick up, each ant gradually increases its radius of perception by one so that it can recognize more dissimilarity in its larger neighborhood. As a result, the probability of picking up increases. The radius increases as long as the ant is unable to perform pickups, and the radius does not hit a maximum threshold. When the radius threshold is hit, the ant reverses the process. That is, the radius gradually decreases depending on ant's ability to stimulate pickups until it reaches zero, at which point the ant is terminated.

To sum up, while ants are moving, increasing/decreasing ant radius of perception stimulates ants to pick up/drop off objects. The gradual increase of radius maximizes inter-cluster dissimilarity because it increases spatial locality among clusters. The gradual decrease of radius of perception, by contrast, maximizes intra-cluster similarity because it decreases spatial locality among the objects within the same cluster. There are many benefits for this strategy: (i) it makes much broader exploration of the solution space as the probability of ant pickup/drop-off increases; (ii) it enables ACA-AR to converge the optimal number of clusters most of the time because the ants can balance the intra-cluster similarity and inter-cluster dissimilarity; (iii) it makes the algorithm more capable of detecting data outliers; (iv) it substantially improves the spatial separation of clusters on the grid which is an essential requirement to retrieve the clusters.

3.2.3 Termination condition

In the existing ACA models, the algorithm terminates when the ants reach a predefined number of maximum steps (iterations) that is determined by trial and error. In ACA-AR, the ant is terminated after using all possible values for the radius (i.e., termination condition of ant depends on the value of ant radius). Each ant is initialized with $r = 1$. When the ant becomes unable to pick up objects, it increases its radius by 1. The increase of radius continues as long as the radius is less than a threshold ($r \geq \frac{1}{4}\sqrt{grid\ area}$). The radius threshold is determined ,experimentally, to enable each ant detecting the maximum dissimilarity in a given set of objects when there exist at a least two clusters in the data. When the ant reaches the ant's maximum threshold, the ant reverses the process. That is, the ant gradually decreases the radius each time it becomes unable to pick up objects until it reaches zero, at which point the ant is terminated (i.e., the ant is terminated after using all possible values for the radius). When an ant is terminated, it is removed from the grid.

A high-level description of ACA-AR including the aforementioned modifications is illustrated in Algorithm 1.

Algorithm 1 ACA with Adaptive Radius of Perception

Input: feature vectors $\{x^i\}_{i=1}^n$ $x^i \in R^m$ N_{ants} $grid_{(h \times w)}$ max radius r M_{steps}

Output: grid coordinates $\forall x^i \in \{x^i\}_{i=1}^n$

```
1:  $\forall x^i \in \{x^i\}_{i=1}^n$ , assign $x^i$ to a random grid location
2:  $\forall$ $ant_i \in N_{ants}$, assign $ant_i$ to a random grid location
3:  for $ant \in N_{ants}$ in parallel do {main loop}
4:    $ant.radius \leftarrow 1$ {initialize ant radius of perception}
5:    while $ant.radius > 0$ do
6:      $ant.pickups \leftarrow 0$ {this is to count the number of pickups that are performed
        successfully by the ant}
7:      for $step \in \{1, ..., M_{steps}\}$ do
8:        if $ant.location \neq null$ then {there is object at ant's location}
9:          if $ant$ $is$ $unladen$ then {ant does not carry object}
10:           obj $\leftarrow$ grid(ant.location) {get the object using ant position}
11:           draw a random number $R \in [0, 1]$
12:           compute $P_{pickup}(obj)$, equation (3.5)
13:           if $P_{pickup}(obj) \geq R$ then {perform pick up}
14:             ant.obj = grid(ant.location) {ant carries object}
15:             grid(ant.location) $\leftarrow$ null {remove object from grid}
16:             $ant.pickups \leftarrow ant.pickups + 1$
17:           end if
18:         else{ant is loaded with object so it continue wandering}
19:           move ant to a random grid location
20:           continue {go to while loop}
21:         end if
```

```
22:         else{there is no object at ant's location}
23:           if ant is carrying object then
24:             obj← ant.carrying
25:             draw a random number $R \in [0, 1]$
26:             compute $P_{drop}(obj)$, equation (3.6)
27:             if $P_{drop}(obj) \geq R$ then {perform drop off}
28:               grid(ant.location) ← ant.obj {store ant's object in the grid}
29:               ant.obj = null {ant becomes unladen}
30:             end if
31:           else{ant is unladen, and grid location is empty so ant continue wander-
    ing}
32:             move ant to a random grid location
33:             continue   {go to while loop}
34:           end if
35:         end if
36:       end for
37:       increase ant radius by 1 as long as ant radius does not reach max radius $r$
38:       decrease ant radius by 1 if ant radius $r$ reaches max radius $r$
39:     end while
40: end for
41: $\forall x^i \in \{x^i\}_{i=1}^{n}$  retrieve grid coordinates of $x^i$
42: identify clusters using cluster retrieval algorithm 2
```

3.3 ACA-AR with Pairwise Constraints

In applications where constraints exist between data, ACA-AR is not enough. The most important operations, pick up and drop-off of objects are restricted by the Must-Linked (ML) and Cannot-Linked (CL) constraints. In this section, I explain how I modify ACA-AR algorithm to address this issue.

In Figure 3.4, ants tend to perform more pickups when there are more dissimilar objects within ants' neighborhood, and they tend to perform more drop-offs when there are more similar objects within ants' neighborhood. I extend this same intuition to incorporate pairwise constraints. The higher the constraint satisfaction within the ant's neighborhood, the greater the drop-off probability. On the contrary, the higher the constraint violation within the ant's neighborhood, the greater the pick-up probability. This intuition can be formulated in two steps. First, we make each ant count the number of satisfied ML and CL constraints using an indicator function that adds one for every satisfied ML or CL constraint as shown in equation 3.7). Second, we add this total number of satisfied constraints to the initial value of $c = 1$, as shown in equation 3.8. As c increases, the probability of drop-off increases and vice versa, as described in equations (3.6 and 3.7).

$$f(x_i) = \sum_{(x_i,x_j)\in ML} \mathbb{1}[l_i = l_j] + \sum_{(x_i,x_j)\in CL} \mathbb{1}[l_i \neq l_j] \tag{3.7}$$

$$c = c + f(x_i) \tag{3.8}$$

In equation (3.7), the term $\mathbb{1}[l_i = l_j]$ is an indicator function that adds one for each satisfied ML constraint within ant neighborhood. Similarly, the term $\mathbb{1}[l_i \neq l_j]$

is an indicator function that adds one for each satisfied CL constraint within ant neighborhood. The flowchart shown in Figures 3.5 and 3.6 illustrates the algorithm steps including the above modifications.

3.4 Cluster Retrieval

Cluster retrieval algorithm assigns a cluster label to each data point. As shown in Figure 3.7, all data instances that belong to the same cluster corresponds to a blob on the grid. In the experiments, I use edgeless grid (torus) which means that data points at the edges are adjacent. To retrieve the clusters, I map data instances into an undirected graph $G(V, E)$ based on their grid coordinates such that $V = \{x_i \in x_{i=1}^{n}\}$. To determine connectivity of G, each data point x_i at position (x, y) is connected to its neighbors at positions $(x \pm 1, y \pm 1)$. As a result, each blob on the edgeless grid forms a connected component in G. Finding connected components is straightforward and can be found using either breadth or depth first search algorithms. The cluster retrieval process is illustrated in Algorithm 2.

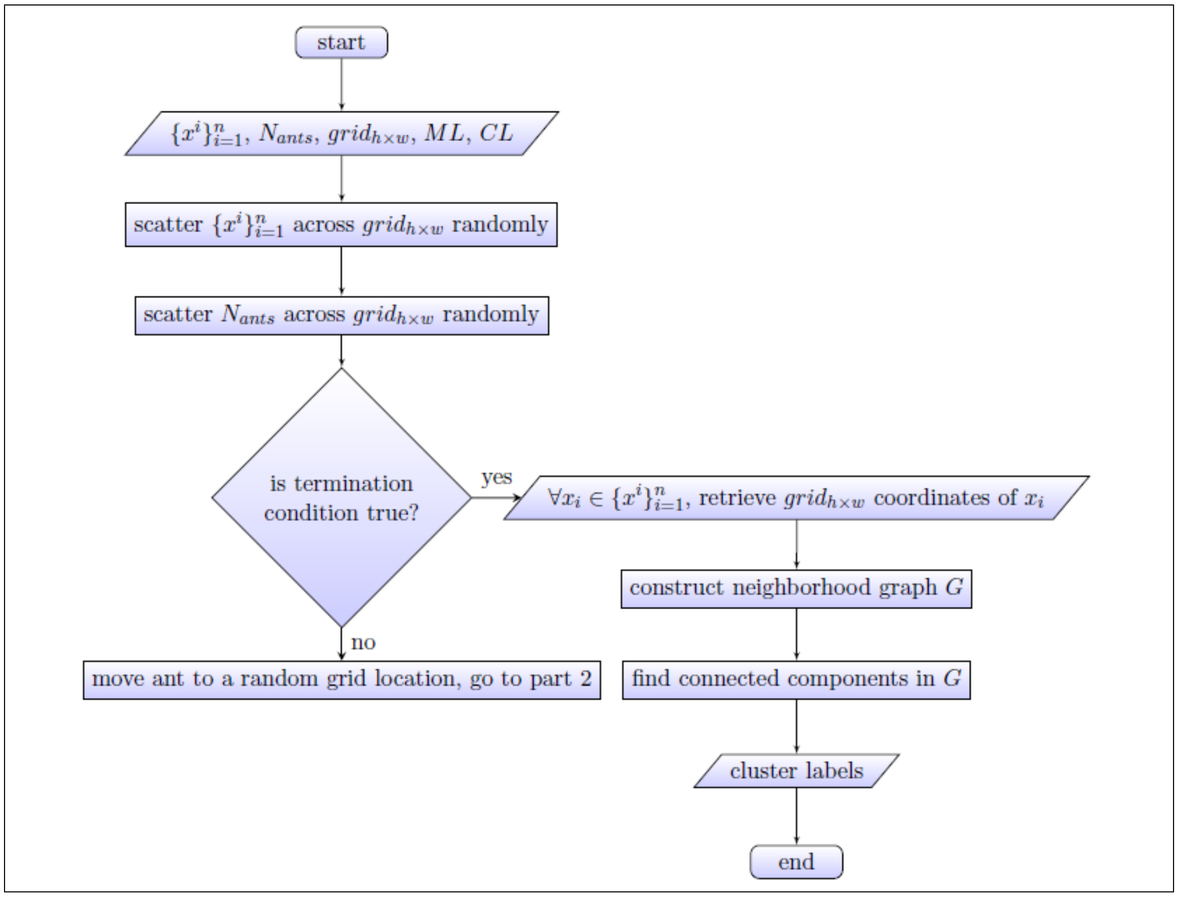

start
$\{x^i\}_{i=1}^n$, N_{ants}, $grid_{h\times w}$, ML, CL
scatter $\{x^i\}_{i=1}^n$ across $grid_{h\times w}$ randomly
scatter N_{ants} across $grid_{h\times w}$ randomly
is termination condition true?
yes
$\forall x_i \in \{x^i\}_{i=1}^n$, retrieve $grid_{h\times w}$ coordinates of x_i
construct neighborhood graph G
find connected components in G
cluster labels
end
no
move ant to a random grid location, go to part 2

Figure 3.5: Modified ACA Flowchart: Part 1

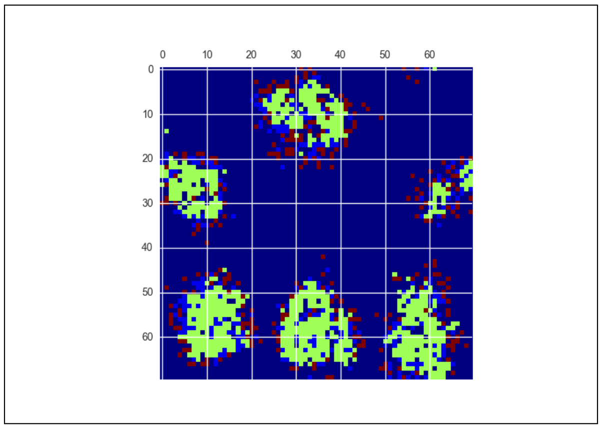


Figure 3.7: Data points that refer to one cluster form well-separated blob. The most dense regions (green interior points within a blob). Red data points determine blob boundaries

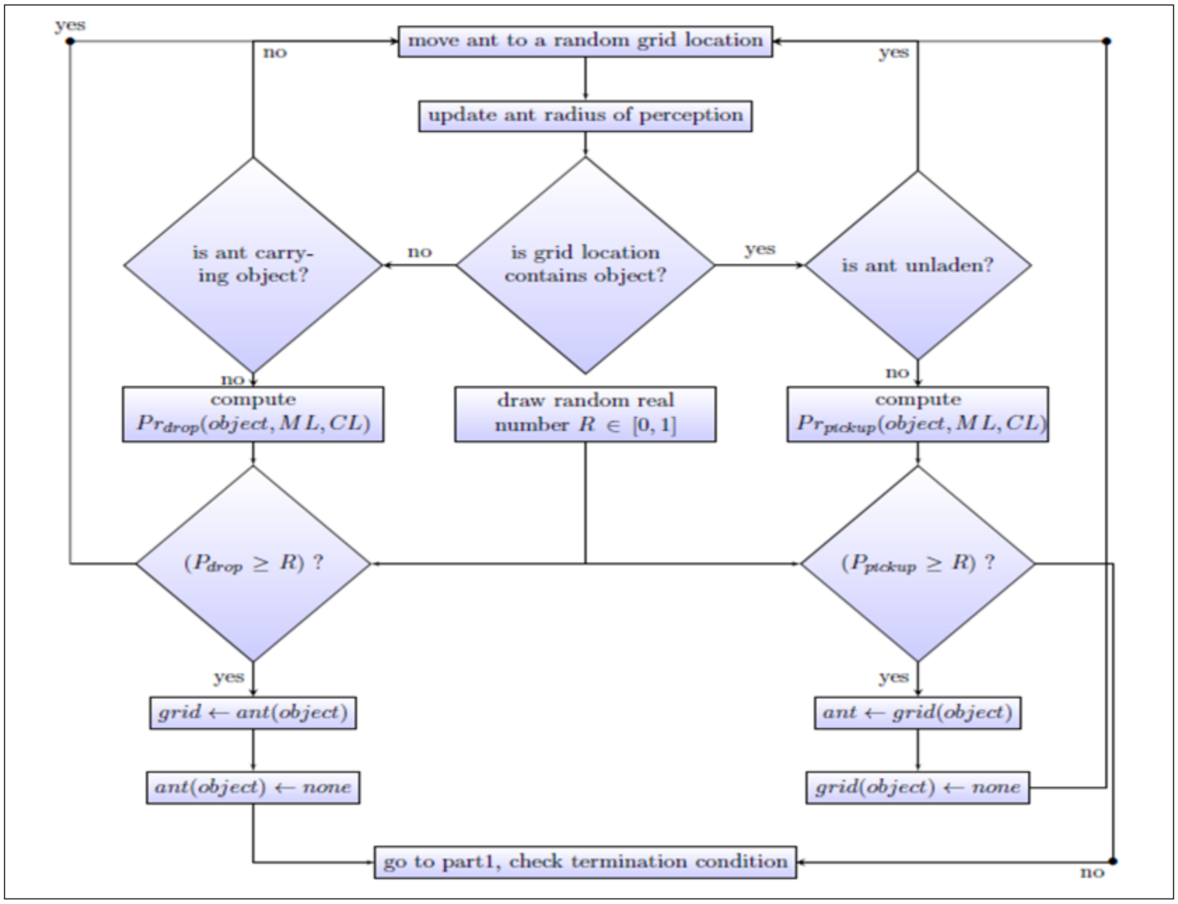

move ant to a random grid location
update ant radius of perception
is ant carrying object?
is grid location contains object?
is ant unladen?
yes
no
compute $Pr_{drop}(object, ML, CL)$
draw random real number $R \in [0,1]$
compute $Pr_{pickup}(object, ML, CL)$
$(P_{drop} \geq R)$?
$(P_{pickup} \geq R)$?
$grid \leftarrow ant(object)$
$ant(object) \leftarrow none$
$ant \leftarrow grid(object)$
$grid(object) \leftarrow none$
go to part1, check termination condition

Figure 3.6: Modified ACA Flowchart: Part 2

Algorithm 2 Cluster Retrieval

Input: grid coordinates $\forall x^i \in \{x^i\}_{i=1}^n$

Output: cluster label l_i $\forall x^i \in \{x^i\}_{i=1}^n$

```
1: initialize undirected graph $G(V, E)$ $V = \{x^i\}_{i=1}^n$
2: for $x_i \in \{x_i\}_{i=1}^n$ do
3:     for $x_j \in neighbors(x_i)$ do
4:         $E = E \cup \{(x_i, x_j)\}$
5:     end for
6: end for
7: identify connected components in $G$ using depth or breadth first search
8: assign a cluster label for each subset of vertices identified as connected component
```

3.5 Algorithm Validation

3.5.1 Benchmark Datasets and Evaluation Metrics

I evaluate the clustering quality of ACA-AR with and without constraints on three benchmark datasets: Iris (Fisher [1936]), Yeast (Horton et al. [2007]), and a subset of the 20 Newsgroups (Nigam et al. [2000]). Each dataset presents a different clustering challenge that can be solved with the proposed algorithm. All datasets are available from the University of California, Irvine machine learning repository (UCI) (Lichman [2013]).

- Iris: consists of 150 instances and 4 features. Iris involves three non-spherical classes: Versicolor, Virginica, and Setosa. The first two classes are overlapped (non-linearly separable) while the third is linearly separable from them. The default similarity measure between two instances is the Euclidean distance.
- Yeast: consists of 10 highly-unbalanced classes that represent the localization site of protein. These classes include 1484 instances where each instance is represented by 8 continuous features. The Euclidean distance is the default similarity metric.
- 20 Newsgroups: This data set is comprised of 20000 documents taken from 20 Usenet newsgroups. It is commonly used for text classification and clustering applications. To highlight the impact of constraint incorporation in ACA-AR, I select 1500 documents from 5 classes which are graphics, misc, PC hardware, Mac hardware, and Windowsx. The classes are highly overlapped and closely

related. I apply the vector space model in order to transform the selected documents into feature vectors. The terms are weighed using $TF-IDF$ scheme, Term FrequencyInverse Document Term Frequency, weighting scheme. The number of features is truncated to 5000, and vector distance is computed by the Cosine similarity.

There are several metrics for external cluster evaluation. Since the ground truth is known for each selected dataset, I adopt the following metrics to evaluate the different clustering quality of ACA with adaptive radius:

- V-measure Hirschberg and Rosenberg [2007]: is an entropy-based measure that measures how successfully the data points that belongs to one class are all included in the same cluster. V-measure is computed as the harmonic mean of different homogeneity and completeness scores. Homogeneity indicates whether each cluster contains only data points of a single class, whereas the completeness indicates how many data points of a given cluster are assigned to the same cluster.

- Adjusted Random Index (ARI) Hubert and Arabie [1985]: The Random Index measures the agreement between two clustering solutions by considering all pairs of data points. It counts how many pairs are correctly clustered relative to the total number of the pairs. ARI adjusts the rand index to vary between -1 and 1 according to expectation with 1 being a perfect match.

- Silhouette Coefficient (SC) Rousseeuw [1987]: is an internal evaluation metric $SC \in [-1, 1]$ with the higher score indicating better defined clusters. Good defined clusters are those where the data points in one cluster are close to each other compared to their next closest cluster.

I compare the clustering quality of ACA-AR with three baselines: k-means, Mean Shift and ACA with constant radius.

- k-means: is a clustering algorithm that partitions a set of data points X into k disjoint clusters C. k-means requires the number of clusters k to be specified. Each cluster is described by its centroid μ, which represents the mean of data points within the cluster. k-meanslocally minimizes a sum-of-squares criterion within each cluster $\sum_{i=1}^{n} \min_{\mu_j \in C} \left\| x_j - \mu_i \right\|^2$. k-means is the baseline for result comparison.

- Mean Shift (MS) (Fukunaga and Hostetler [1975]): is a non-parametric iterative clustering method that does not make any prior assumptions about the number or the shape of clusters. The MS assumes that the data points are sampled from a probability density function (pdf). Therefore, it considers the local-maxima points in the pdf as dense regions (clusters). The intuition of MS is to associate each data point with the nearby local maxima. For each data point, MS fixes a window around it and computes the mean of the data points within that window. Then, it shifts the center of the window to the mean and iterates until it converges. The goal of each iteration is to shift the window towards a denser region of the dataset.

  Given a data point x_i for iteration t, x_i is shifted by the equation $x_i^{t+1} = x_i^t + m(x_i^t)$ where $m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}$. $N(x_i)$ is the neighborhood of data points within a given distance around x_i. The size of neighborhood is determined by the value of bandwidth h in a kernel k. m is the mean shift vector that is computed for each data point and points towards a region of the maximum increase in the density of points (scikit-learn developers).

3.5.2 Clustering Results of ACA-AR

Table 3.1: Cluster Validity - Iris

	k-means	MS	ACA	ACA-AR
V-measure	0.66	0.73	0.64	0.82
ARI	0.62	0.57	0.62	0.71
SC	0.46	0.58	0.43	0.54
number of clusters	3.00	2.00	3.00	3.00

Table 3.1 shows the comparative results of Iris clustering solutions obtained by k-Means, mean shift, ACA with fixed radius, and ACA with Adaptive Radius (ACA-AR). As the metrics indicate, ACA-AR outperforms all other clustering algorithms in identifying the Iris clusters. Mean shift identifies only 2 out of the 3 existing clusters because it fails to separate the overlapped Iris classes. ACA and ACA-AR, in contrast, identify all of the three classes. In addition, ACA-AR gains 18% increase in ARI, as indicated by its V-measure score (82%), compared to ACA. This increase is due to the improvement of the completeness and homogeneity of clusters.

I also evaluate the clustering quality of ACA-AR on the Yeast dataset, which contains a larger number of classes (10) with highly unbalanced class distribution

Table 3.2: Cluster Validity - Yeast

	k-means	MS	ACA	ACA-AR
V-measure	0.53	0.73	0.77	0.82
ARI	0.28	0.57	0.65	0.69
SC	0.35	0.58	0.55	0.57
number of clusters	10.00	2.00	3.00	5.00

Table 3.3: Cluster Validity - 20 Newsgroups

	k-means	MS	ACA	ACA-AR
V-measure	0.21	0.57	0.55	0.61
ARI	0.18	0.49	0.55	0.59
SC	0.04	0.51	0.55	0.57
number of clusters	5.00	3.00	3.00	5.00

(463, 429, 244, 163, 51, 44, 37, 30, 20, 5). Table 3.2 shows the experimental results on this dataset. k-means converges to a local-minima solution with ARI of 28% due to the poor selection of initial centroids. Although mean shift achieves higher scores for ARI and V-measure, it fails to detect classes with small sizes. ACA and ACA-AR, by contrast, outperform mean shift because they are more capable of detecting the small classes.

Clustering 20 Newsgroups documents is challenging because of the "curse of dimensionality", which the result of the fact documents are represented using sparse high-dimensional feature vector (e.g 10000 features). To obtain the results using the cosine distance, I use the spherical k-means, which is a k-means variant for text clustering. To handle the cosine similarity in mean shift, ACA and ACA-AR, I adapt equation (3.1) to $N_h(o_i) = \{o_j : \left\| cosine(\vec{o_i}, \vec{o_j}) \right\| \geq h\}$ as proposed by (Senoussaoui et al. [2013]). Reviewing the results in Table 3.3, k-means obtains the worst results because of class non-linearity and initial selection of cluster centroids. Both mean shift and ACA identify only three out of the five existing clusters because they fail to separate some of the overlapped classes. ACA-AR, by contrast, identify all of the five classes.

3.5.3 Clustering Results of ACA-AR with pairwise Constraints

I evaluate ACA-AR with randomly-generated constraints using 5-fold cross validation as described in (Pourrajabi et al. [2014]). Each dataset is divided into 5 folds where 4 of them are used to create constraints. The entire dataset is clustered, and the left-out fold is used for evaluation. This process is repeated 5 times so that each fold is used for constraint creation and evaluation. The ARI is calculated only on the test set, and the results are averaged over 10 runs of 5 folds. As shown in Figure 3.8, the accuracy (measured by ARI) of the three datasets improves when more constraints are added. Initially (i.e., when no constraints are provided), the ARI score is for ACA-AR (entirely unsupervised).As the number of constraints increases (e.g., 200, ..., 1000), the ARI also increases as a result of using constraints. In this case, the ACA-AR is a semi-supervised algorithm.

For the Iris and Yeast, the accuracy rapidly increased with larger sets of constrains with respect to the unconstrained accuracy. However, since the classes of 20 Newsgroups are highly overlapped, it shows a slower increase in accuracy with more constraints.

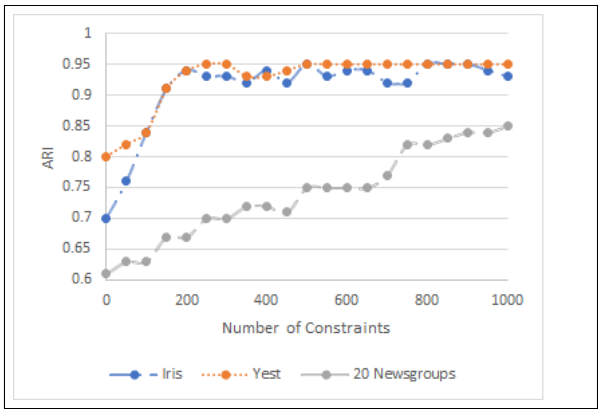


Figure 3.8: ARI Curves: Constraints vs. Accuracy (ARI)

Chapter 4

Case Study: Aspect Based Sentiment Analysis

In this chapter I explain aspect based sentiment analysis. I will test the feasibility of my proposed clustering algorithm on this problem.

4.1 Introduction

During 2012 USA presidential election, Topsy Labs, a Twitter partner for social media analytics, launched a new feature *Twitter Political Index*, which is a daily sentiment meter that measures the level of public sentiment expressed by Twitter users about presidential candidates. This meter assigns each candidate with a daily score between 0 and 100, with 100 being the most positive. These scores are calculated by extracting sentiments from 2 million Twitter posts a week. On November 7th, the day after the election, for example, Obama scored 85 and Romney scored 57 (Moore

[2012]). This new style of extracting public sentiments from written source materials such as social-media postings, product reviews, blogs and news articles is the main task of *sentiment analysis* (SA). During the last decade, SA has been applied not only to politics but also to many other disciplines, including stock markets, voting-based TV shows and movie-revenue predictions. Despite its popularity, SA still poses intellectual and practical challenges at both the academic and industrial levels (Liu [2015]; Liu [2010]; Pang and Lee [2008]; Vohra and Teraiya [2013]).

SA is a significant discipline of research because our decision-making is social. Individual's decision-making is highly influenced by the opinions of their peers. For instance, people usually look for their peers online opinions to decide whether or not to buy a product, visit a specialist or vote for a candidate. Similarly, companies need to develop insights about their products or services by identifying customer opinions. These insights are vital to improve marketing decisions, campaign success and product messaging. Such needs, besides the massive amount of opinionated text at our fingertips, have led to the emergence of the field of SA.

4.2 Sentiment Analysis

Sentiment Analysis, also known as *Opinion Mining* (OM), refers to the computational study of how subjective information are expressed in textual units. SA and OM are synonymous and hence often used interchangeably. SA aims at identifying, extracting and aggregating the attitudes of speakers or writers toward topics or entities and their aspects or features. SA also aims at evaluating and classifying the intensity of such attitudes. These challenging tasks incorporate the use of Natural Language Processing (NLP), Machine Learning (ML), Computational Linguistics (CL) and Data Mining (DM).

There are two main levels of SA: *coarse-grained and fine-grained*. Coarse-grained SA aims at determining whether the attitude is positive or negative at the document or sentence level. It assumes that a document (e.g. product review) or a sentence expresses a single opinion on a single entity. On the other hand, fine-grained SA aims at pinpointing where in a sentence an opinion is expressed, what are the opinion polarity and intensity, and what the opinion is directed towards. This fine-grained level of SA is also known as *Aspect-based SA* (ABSA). In this thesis, I investigate ABSA in the domain of product reviews.

4.3 Aspect-based Sentiment Analysis (ABSA)

Unlike coarse-grained SA, the goal of ABSA is to identify sentiments expressed towards different features or aspects of entities. For instance, the sentence "The food was delicious but the service was awful." involves two contradicting sentiments *(i.e. positive and negative)* on two different aspects *(i.e. food and service)*, respectively. Although coarse-grained SA has proven beneficial for individuals and businesses, it is still insufficient for most applications. In the domain of product reviews, for example, it is very common that consumer expresses different sentiments *(e.g. delicious, awful)* towards different product aspects *(e.g. food, service)* within a single sentence. These sentiments may vary in polarity and intensity from one product to another or from one aspect to another. Also, a consumer may favor a product because s/he likes some aspects of the product. Therefore, businesses need to identify from online reviews what product aspects consumers like or dislike and to what extent. In short, there is a need for ABSA to recognize, aggregate and summarize such material in an informative way.

Furthermore, ABSA has become a prerequisite to support new applications in the domains of Information Retrieval (IR) and Natural Language Processing (NLP) such as opinion spam detection, opinion helpful estimation or comparative sentence opinion mining as shown in Figure 4.1. This need has emerged as a result of the rapid increase in online user-generated contents, particularly in product reviews. According to a survey conducted by comScore, Inc. (NASDAQ: SCOR) Kelsey Group on more than 2000 respondents (Liu [2015]), 32% of the participants have provided online ratings for a service, product or person. Also, 20% of the participants have searched reviews on a daily basis. Moreover, the study reveals that consumers are willing to pay at least 20% more for 5-star service than 4-star service. In the discipline of information retrieval, for instance, there is a demand for opinion-oriented search engines that can retrieve relevant documents to opinionated queries. Likewise, in natural language processing, there is a need for question-answering systems that can provide answers to opinion-based questions. In addition to the aforementioned applications, there are a wide range of domains whereby ABSA can be employed and for valuable reasons. Such domains include business intelligence, stock market prediction, recommendation systems and text-to-speech engines (Pang and Lee [2008]).

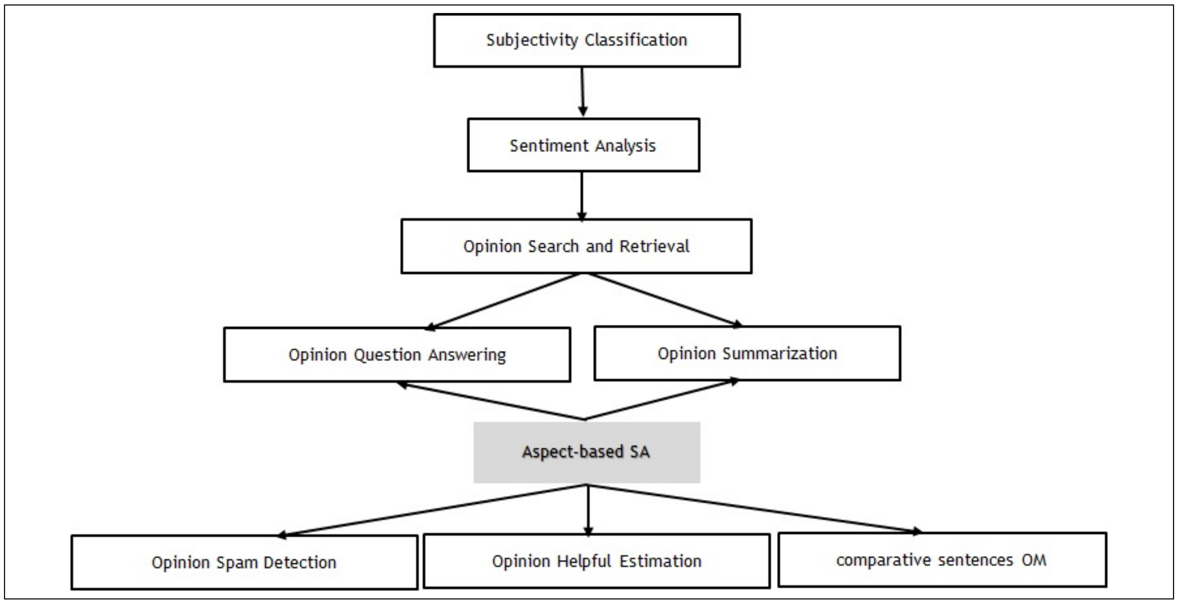

Subjectivity Classification
Sentiment Analysis
Opinion Search and Retrieval
Opinion Question Answering
Opinion Summarization
Aspect-based SA
Opinion Spam Detection
Opinion Helpful Estimation
comparative sentences OM

Figure 4.1: ABSA is a prerequisite to other NLP and IR tasks

4.3.1 ABSA Formal Definition

Formally, given a corpus C of product reviews, suppose that:

1. E_i: an entity i
2. A_{ij}: an aspect j of entity i
3. H_k: opinion holder
4. T_l: sentiment time
5. S_{ijkl}: sentiment polarity/intensity S_i expressed on an aspect A_j of entity E_k by a holder H_k at time T_l

The problem of aspect-based sentiment analysis is to identify all opinion quintuples $(E_i, A_{ij}, S_{ijkl}, H_k, T_l)$ for each review in C such that the five components correspond to each other; any mismatch is an error (Liu [2010]).

Based on this definition, the major task of ABSA can be divided into four subtasks as illustrated in Figure 4.2 (Pontiki et al. [2014]). Given a corpus of reviews, the first subtask aims at identifying all single or multi-word terms that refer to particular aspects of the target entity (e.g. laptops, restaurants, etc.) within the corpus. For example, in the sentence "The *staff* and the *service* are amazing, but not the *food*.", the aspect terms are *staff*, *service* and *food*. In the second subtask, predict aspect term polarity, each identified aspect term is assigned a sentiment polarity (e.g. positive, negative and neutral) with respect to sentiment expressed in the sentence. For instance, the aspect terms *staff*, *service* and *food* will be assigned *positive*, *positive* and *negative* polarity, respectively. The goal of the third subtask is to group aspect terms that refer to the same aspect or entity into aspect categories. For example, in the sentence "The *menu* is great, but the restaurant is *expensive*." the aspect term *menu* refers to aspect category *food*, whereas *expensive* refers to aspect category *price*. In the fourth subtask, each aspect category (e.g. food, price, service, etc.) is assigned an overall sentiment score based on its content (i.e., aspect terms and their sentiment polarity). Finally, both aspect terms and aspect categories can be used by an aspect based sentiment analysis system to generate a structured summary as the one shown in Figure 4.3.

4.3.2 Challenges: Why ABSA is hard?

ABSA presents many challenges. These challenges can be defined based on the two major tasks of ABSA as follows:

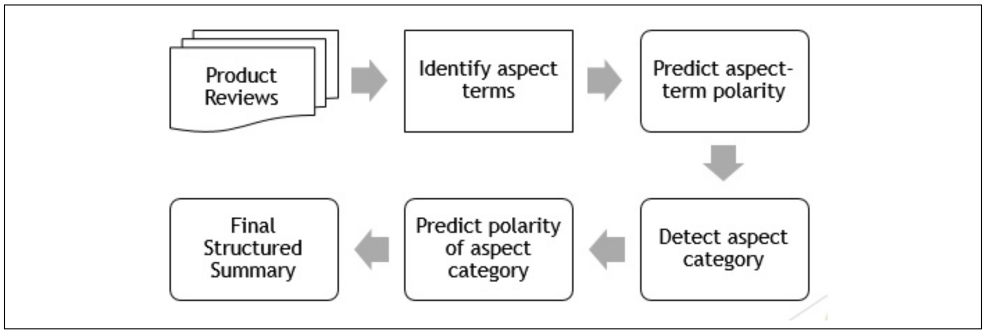
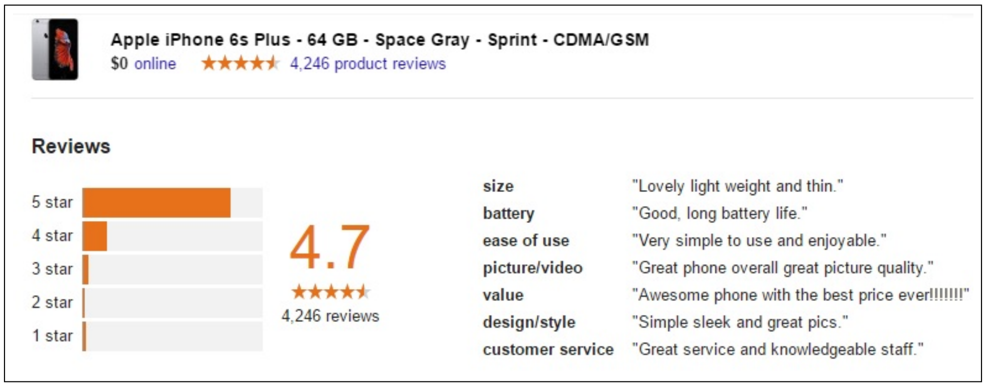


Figure 4.2: Subtasks of ABSA


Apple iPhone 6s Plus - 64 GB - Space Gray - Sprint - CDMA/GSM
$0 online ★★★★★ 4,246 product reviews
Reviews
5 star
4 star
3 star
2 star
1 star
4.7
4,246 reviews

size	"Lovely light weight and thin."
battery	"Good, long battery life."
ease of use	"Very simple to use and enjoyable."
picture/video	"Great phone overall great picture quality."
value	"Awesome phone with the best price ever!!!!!!!"
design/style	"Simple sleek and great pics."
customer service	"Great service and knowledgeable staff."

Figure 4.3: Structured Summary Output

1. How to identify sentiment target?

   Sentiment can be directed toward entities or aspects of entities. Identifying sentiment targets is challenging because the entity or any of its aspects can be described using variable-length text units (e.g. word, phrase or sentence). This description can be explicit (i.e., entity or aspect is literally mentioned in text) or implicit (i.e., non-literal where entity or aspect has to be inferred from text).

2. How to predict sentiment per target?

   Sentiment prediction is to determine the polarity of the sentiment source with respect to predetermined categories (e.g. positive, negative and neutral), and evaluate its intensity with respect to a predefined scale (e.g. 1 to 10). Also, sentiment can be expressed using variable-length text units either literally or non-literally. However, predicting sentiment is more challenging because it can be expressed using a wide variety of linguistic constructs and expressions. In addition, sentiment is not only domain dependent but also context dependent.

Therefore, the problem of ABSA involves two main tasks: *aspect identification and sentiment prediction.* The approaches/methods involved in solving these two tasks are explained in sections (4.5; 4.6). In this thesis, I investigate both tasks using constrained clustering based on swarm intelligence.

4.3.3 Constrained Clustering to ABSA

There exist two main reasons that make constrained clustering a feasible approach to ABSA. First, labeled data for ABSA is unavailable in many domains and labeling data is costly and time-consuming. This makes applying standard classification algorithms for ABSA infeasible. However, ABSA makes available large amounts of unlabeled data as well as small amounts of supervisory information that indicate whether particular data objects (text units) are similar or dissimilar to be grouped in the same cluster or not. The supervisory information can be naturally encoded in the form of instance-level pairwise constraints. Second, ABSA is highly domain-dependent and subjective. Therefore, imposing pairwise constraints on ABSA clustering is essential as they reflect user's perspective and knowledge on text similarity in the domain.

Constrained clustering aims at incorporating such supervisory information to guide a clustering algorithm towards solutions with minimally-violated constraints (Wagstaff and Cardie [2000]; Basu et al. [2008]).

To formalize the problem of ABSA as a constrained clustering problem, I assume the following pre-processing steps:

- Product reviews are given in a semi-structured format, such as Extensible Markup Language (XML) or JavaScript Object Notation (JSON). Therefore, some opinion components such as entity E_i, opinion holder H_k and sentiment time T_l are explicitly identified.

- Each review is tokenized into text units which involve sentences, phrases and words. This tokenization is essential because a product aspect A_{ij} can be embedded in a single word, a phrase or an entire sentence. Similarly, a sentiment can be expressed in a single word, a phrase or an entire sentence.

- Each text unit is represented n-dimensional feature vector.

- The semantic similarity between two text units can be quantified by measuring the distance between their corresponding feature vectors. For example, a greater value of cosine similarity distance implies stronger semantic similarity. Other common distance measures include Euclidean distance, Minkowski, and Manhattan distance.

- There exists limited annotated dataset (Pontiki et al. [2015]) available for ABSA, while large amounts of unlabeled product reviews for ABSA are also accessible (e.g. Amazon Product Reviews, (McAuley and Leskovec [2013]).

Let $V = \{v_1, v_2, v_3, ..., v_n\}$ be the set of feature vectors that represent text units, $ML = \{(v_i, v_j),\ v_i, v_j \in V\}$ be a must-linked set of data pairs where a pair (v_i, v_j) implies that v_i and v_j are known to be in the same cluster, and $CL = \{(v_i, v_j),\ v_i, v_j \in V\}$ be a cannot-linked set of data pairs such that v_i and v_j are known to be in different clusters.

The first task in ABSA is the *aspect identification*, identifying aspect categories A_{ij}, can be accomplished by grouping all text units (aspect terms) that refer to a particular aspect category, A_{ij}, in the same cluster. Therefore, aspect identification can be tackled by partitioning V into a set of k-disjoint clusters $C = \{c_1, c_2, c_3, ...c_k\}$ such that the intra-cluster semantic similarity is maximized, the inter-cluster semantic similarity is minimized, and both types of pairwise constraints are maximally satisfied. Similarly, the second task in ABSA is to *predict a sentiment class for each identified aspect cluster*. Likewise, this prediction can be achieved by grouping text units into k disjoint clusters such that the text units in one cluster have the same sentiment class. Formally, we need to partition V into a set of k-disjoint clusters $C = \{c_1, c_2, c_3, ...c_k\}$ such that the intra-cluster sentiment similarity is maximized, the inter-cluster sentiment similarity is minimized, and the given constraints are minimally violated.

Since both tasks of ABSA can be defined as a constrained clustering problem, the problem turns out to be finding an optimal or near-optimal clustering solution C^* with respect to a fitness function (similarity function). However, finding such C^* is *NP-hard* problem (Welch [1982]) because the number of feasible solutions grows exponentially with respect to the number of data items (vectors) to be clustered. Suppose we have n text units such that each text unit is mapped into a feature vector of dimension d, and the number of desired clusters is k, the number of feasible solutions is given by equation 4.1,

$$N(n,k) = k^n \tag{4.1}$$

Therefore, heuristics are needed to solve the clustering problem. In this thesis, I consider ant based clustering technique discussed in chapters 2 and 3.

4.4 Essential Approaches to ABSA

As we mentioned earlier in the example, “The food was delicious but the service was awful.”, the goal of ABSA is to identify *(food and service)* as aspect terms, and to predict that *(delicious and awful)* hold *(positive and negative)* sentiments toward these aspects, respectively. Hence, in ABSA, there are essentially two main sub-tasks. The existing approaches for tackling ABSA can, therefore, be divided into two major categories, as described in Figure 4.4. For the first category, *aspect identification*, there are four main methods, described in section 4.5. For the second category, *sentiment prediction and analysis*, there are mainly two methods, supervised machine learning and lexicon-based, described in section 4.6.

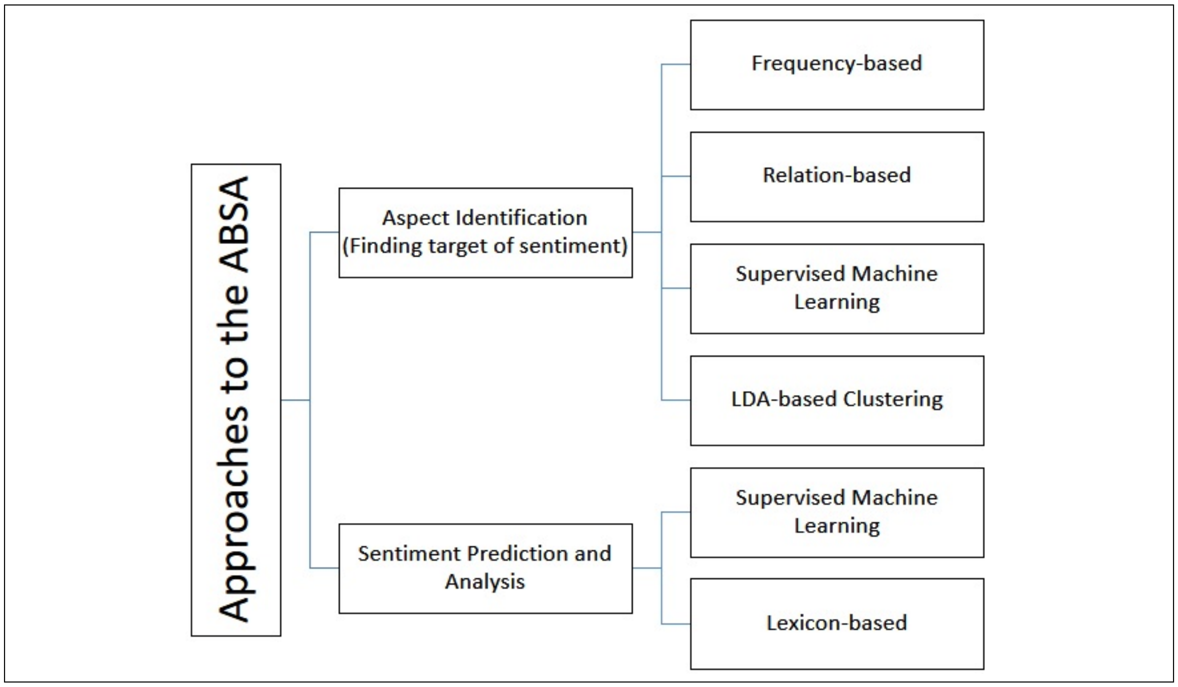

Approaches to the ABSA
Aspect Identification (Finding target of sentiment)
Sentiment Prediction and Analysis
Frequency-based
Relation-based
Supervised Machine Learning
LDA-based Clustering
Supervised Machine Learning
Lexicon-based

Figure 4.4: Essential Approaches to ABSA

4.5 Approaches to Aspect Identification

There are four essential methods for aspect identification. However, there exist other hybrid methods that are proposed by combining two or more existing methods. The basic idea of such an approach is to combine the advantages of two essential methods. In the following subsections, the main idea of each method is presented. In addition, the strengths as well as the limitations of each one are provided in Table 4.1.

Table 4.1: Approaches to Aspect Extraction: Pros & Cons

Approach	**Pros**	**Cons**
Frequency-based	Simplicity and efficiency	produce many non-aspects miss low-frequency aspects non-portable manual tuning of various parameters
Relation-based	find low frequency aspects	needs a large variety of syntactic rules pattern matching produces too many aspects miss low-frequency aspects require manual tuning
Supervised Machine Learning	overcome frequency-based limitations by learning parameters from data	need a substantial amount of manually annotated dataset manual inspection and domain expertise are needed domain dependence feature selection problem
Hybrid Methods	limit the number of non-aspects	miss low-frequency aspects require manual tuning
Topic Models (LDA)	unsupervised, aspect extraction and grouping at the same time	relies on bag-of-words model produce topics that are semantically unrelated need a large volume of data topics are unlabeled; need mapping to aspects and entities LDA is designed to work at the document level (topics are broad in scope) while aspects are locally defined in sentences

4.5.1 Frequency-based Methods

Frequency-based method (Hu and Liu [2004]) relies on the assumption that aspects and entities are described explicitly by a limited set of the most frequently-occurring expressions. Such expressions are usually nouns or noun phrases. This method is simple and straightforward, hence it is quite effective and commercially implemented (Liu [2015]). However, a direct shortcoming of this method is that not all frequent expressions always refer to entities or aspects. More importantly, infrequent expressions that describe entities or aspects are strong candidates to be missed by this method. Also, this method is not applicable when the corpus is a mixture of small groups of different product reviews. Consequently, several refinements ([Liu et al., 2005]; Scaffidi et al. [2007]; Li et al. [2009]; [Hai et al., 2011]; [Long et al., 2010]) have been proposed to improve the precision of frequency-based method. For instance, the work in (Hai et al. [2011]) applies association rules to extract implicit aspects from a co-occurrence matrix that combines explicit aspects with sentiment expressions. Another refinement in (Long et al. [2010]) suggests the use of grammatical dependencies to identify infrequent aspects.

4.5.2 Relation-based Methods

Sentiments and their targets are related via many syntactic relations because they both occur together within sentences. Syntax-based methods rely on exploiting these syntactic relations to extract either sentiments or targets (Hu and Liu [2004]). This means that identifying sentiment words and expressions can be used for extracting targets and vice versa. For instance, the adjective modifier relation between the words *delicious* and *food* in the expression *delicious food* can be used to identify that *food* is a target if *delicious* is marked as a sentiment word. Syntax-based method can detect infrequent aspects efficiently; however, to get good coverage, it needs describing a large variety of syntactic rules. Syntax-based methods have been elaborated in (Blair-Goldensohn et al. [2008];Ruppenhofer et al. [2008]; Kobayashi et al. [2007]) to include dependency relations. The basic idea in such elaborations is to generate all syntactic relations from a given corpus using a dependency parser, and then to identify all dependency patterns that relate sentiments and targets. This idea has also been developed into double propagation method (Qiu et al. [2011]), which is a bootstrapping method for simultaneous extraction of sentiments and targets. DP starts from an initial set of sentiment words and applies certain dependency relations to create an initial set of targets. The extracted targets are then used to expand the sentiment set with new sentiment words which, in turn, are used to find new targets. This process continues until both sets converge.

4.5.3 Supervised Machine Learning

Aspect extraction through supervised machine learning (ML) is considered as a sequence-labeling problem, hence it was investigated using Chain-Conditional Random fields (CRF) and Hidden Markov Models (HMM). However, like all supervised ML techniques, both CRF and HMM require the availability of manually annotated training datasets. These datasets comprises of thousands or even millions of examples that are annotated at the word level. In addition, both CRF and HMM stumble due to the feature selection problem. The study in (Jin et al. [2009]) proposes lexicalized HMM's to extract product aspects from reviews whereby linguistic features, such as part-of-speech and surrounding contextual clues of words, are integrated into automatic learning. CRF outperforms HMM for aspect extraction problem (Liu [2015]) because it can handle overlapping features. The work in (Jakob and Gurevych [2010]) investigates the performance of CRF-based approach in single and cross-domains. In this work, different features such as the current word, part-of-speech tag, the word distance and the dependency path are employed.

4.5.4 LDA-based Clustering

Most unsupervised models proposed for aspect extraction are based on topic models, in particular Latent Dirichlet Allocation (LDA) model (Blei et al. [2003]). LDA is a generative statistical model used for topic modeling. It assumes that documents exhibit multiple topics, and each topic is a probability distribution over words. Therefore, the output of LDA model is a set of clusters of words, whereby each cluster represents a topic in a given corpus. LDA is similar to the probabilistic latent semantic analysis ([Hofmann, 1999]), but it assumes that topic distribution is a Dirichlet prior rather than a uniform distribution. Although LDA has been successfully implemented for topic modeling, it has many limitations to be directly implemented in the context of ABSA (Titov and McDonald [2008]). First, aspects or entities are described by semantically-related expressions. LDA, however, produces topics that are semantically-unrelated because it depends on bag-of-words model. Second, LDA generates clusters of unlabeled topics, hence a method is required to map topics to entities or aspects. Third, LDA is designed to work at the document level. This means that it produces topics that are broad in scope, while aspects are locally defined within sentences.

To elaborate LDA for ABSA, (Titov and McDonald [2008]) proposes multi-grained LDA. The idea of multi-grained LDA is to find a global set of topics and a local dynamic set of topics. Global topics are assumed to describe entities while local topics are assumed to describe aspects. To find local topics, the work in (Mei et al. [2007]) treats each document as sliding windows of overlapped sentences. Another similar work (Lu et al. [2011]) performs LDA at the sentence level. LDA has also been combined with sequence models like CRF (Li et al. [2010]) in order to distinguish between aspect words and background words and HMMs (Lakkaraju et al. [2011]). This distinction depends on incorporating syntactic relations between aspects and sentiments.

4.6 Sentiment Prediction and Analysis Approaches

Approaches to sentiment prediction and analysis aim at determining sentiment category or sentiment score for each identified aspect. For example, if an entity is a cell phone, the identified aspects could be size, battery, price, design, etc. Supervised machine learning and lexicon-based approaches are two basic proposed methods in the literature. The former models the problem as a multi-class classification problem, whereas the latter depends on retrieving sentiments from predefined lexicons. However, since each approach has its own shortcomings, a third hybrid approach has also been proposed to alleviate the drawbacks of each one.

4.6.1 Supervised Machine Learning

Sentiment analysis using supervised machine learning has been successfully implemented at the coarse-grained level, but not at the aspect level. This is because the features used at the coarse-grained level are independent of the targets of the sentiments. Examples of such features include the current word, part-of-speech, negation words and whether the word is given in a sentiment lexicon or not. ABSA, by contrast, needs to consider sentiment targets (entities and aspects) in model learning. Therefore, it needs to incorporate features that depend on the targets of the sentiments.

For instance, (Jiang et al. [2011]) generate target-dependent syntactic features using parse tree. However, this approach assumes that aspects and entities are identified in advance. Another approach proposed by (Boiy and Moens [2009]) is to determine the application scope for each sentiment expression.

4.6.2 Lexicon-based Approach

Sentiment lexicons or dictionaries are essential resources for sentiment analysis regardless of its granularity. Each word, phrase or idiom in a sentiment lexicon is tagged with a sentiment class or score. For example, (Liu [2009 (accessed May 5, 2016]) maintains a sentiment lexicon consisting of 2006 positive word and 4783 negative words. Another frequently used Lexicon is the MPQA (Multi-Perspective Question Answering) maintained by (Wiebe et al. [2005]). The most frequently used lexicon is the SentiWordNet (Choi and Cardie [2009]; Baccianella et al. [2010]) which attaches positive and negative real-valued sentiment scores to WordNet synsets. Finally, The Harvard General Inquirer (Stone et al. [1966]) is a lexicon attaching syntactic, semantic, and pragmatic information to part-of-speech tagged words. Sentiment Lexicons are constructed using WordNet propagation algorithm (Hu and Liu [2004]). The algorithm begins with an initial hand-crafted seed-sets of size n, and it then propagates the WordNet relations (synonyms and antonyms) from the hand-crafted seed-sets, thereby expanding their size. The expanded sets at iteration i are used as seed-sets for iteration $i + 1$, generally after removing any pairwise overlap between them.

Using lexicon-based approach to ABSA involves three major steps. In the first step, all sentiment expressions (words and phrases) in each sentence that contains one aspect or more are extracted. These expressions are assigned with polarity score based on sentiment lexicon (e.g. -1 or +1). In the second step, a set of rules for handling language constructs are applied. These rules, for instance, specify how to manipulate sentiment shifters (e.g. contrary constructs and negations). These rules also help infer the polarity of the words or phrases that are not listed in sentiment lexicon. In the last step, an aggregate function is applied to determine the final polarity per aspect.

4.6.3 Hybrid Approach

Neither the supervised machine learning approach nor the lexicon-based approach scores high accuracy in ABSA. The accuracy of the former stumbles as a result of the feature selection problem (bag-of-words). The lexicon-based approach, on the other hand, suffers from ambiguity of words, multilinguality, granularity and the differences in sentiment expressions among textual genres. Hybridization is to combine two or more methods of different approaches to gain the benefits of each one. For instance, many sentiment applications rely on sentiment lexicons to supply features to a supervised classifier (Blair-Goldensohn et al. [2008]). Another work (Blair-Goldensohn et al. [2008]) finds frequent aspects using maximum-entropy classifier, and implements rule-based method to identify infrequent aspects. Similarly, (Raju et al. [2009]) suggests using the Dice similarity as measure to classify noun phrases that are about the same aspect. Given two noun phrases p_i, p_j, the Dice similarity coefficient is defined as $\frac{2|S_i \cap S_j|}{|S_i|+|S_j|}$ where S_i, S_j are the the sets of uni-grams, bi-grams refer to two noun phrases p_i, p_j.

4.7 Drawbacks of ABSA Clustering Techniques

It has been argued that neither conventional clustering methods, such as k-means nor LDA-based clustering methods perform well in ABSA (Zhai et al. [2011]; Vohra and Teraiya [2013]; Liu [2015]). There are two main reasons for this.

First, in ABSA, using appropriate semantic-similarity measure is the key to achieving high-quality clusters. There are two main types of semantic-similarity measures used in ABSA clustering approach, lexical similarity and distributional similarity. Lexical similarity of words relies on using pre-existing knowledge resource (e.g. thesauri, semantic network) to measure the semantic similarity between words. Distributional similarity, on the other hand, depends on learning distributional features of words given their context (Mikolov et al. [2013b]). The technique is based on distributional hypothesis which states that words with similar meanings tend to occur in similar contexts. Although lexical and distributional similarity measures can capture different types of semantic relationships of words (Mikolov et al. [2013b]), they still lack the semantic features about sentiments of words. Therefore, using either distributional or lexical similarity to guide clustering algorithm in ABSA is insufficient.

Second, despite the wide variety of clustering approaches, such as connectivity-based (hierarchical), centroid-based (partitioning or k-means), graph-based (Clique), distribution (Expectation-Maximization), and Density (DBSCAN and OPTICS) (Aggarwal and Reddy [2013]), previous ABSA research have focused on k-means (Zhai et al. [2011]). The k-means algorithm is the most commonly used method because it is easy to implement, and it is efficient in terms of computation time. However, it suffers from convergence to a local optimum, and the results are highly affected by the selection of initial partitions.

To overcome the shortcomings of previous approaches in terms of object heterogeneity, efficiency, simplicity, and scalability, I investigate aspect based sentiment analysis clustering problem through my proposed semi-supervised constrained ant brood clustering approach discussed in the previous chapters.

Chapter 5

CACA-AR to ABSA Tasks

In this chapter I describe how I apply my constrained ant brood clustering with adaptive radius algorithm (CACA-AR) to the two tasks in ABSA: aspect identification and sentiment prediction. Figure 5.1 shows the clustering framework of my approach to predict aspect category and sentiment class at the sentence level. The benefit of this approach is threefold: (i) it takes the advantages of swarm intelligence features such as decentralization, inherent distribution, collaboration, and self-organization; (ii) it enhances the quality and the convergence of clustering by making effective use of the approach to guide the clustering algorithm; (iii) since pairwise constraints reflect domain knowledge, incorporating them in ABSA clustering mitigates ABSA's domain dependency problem.

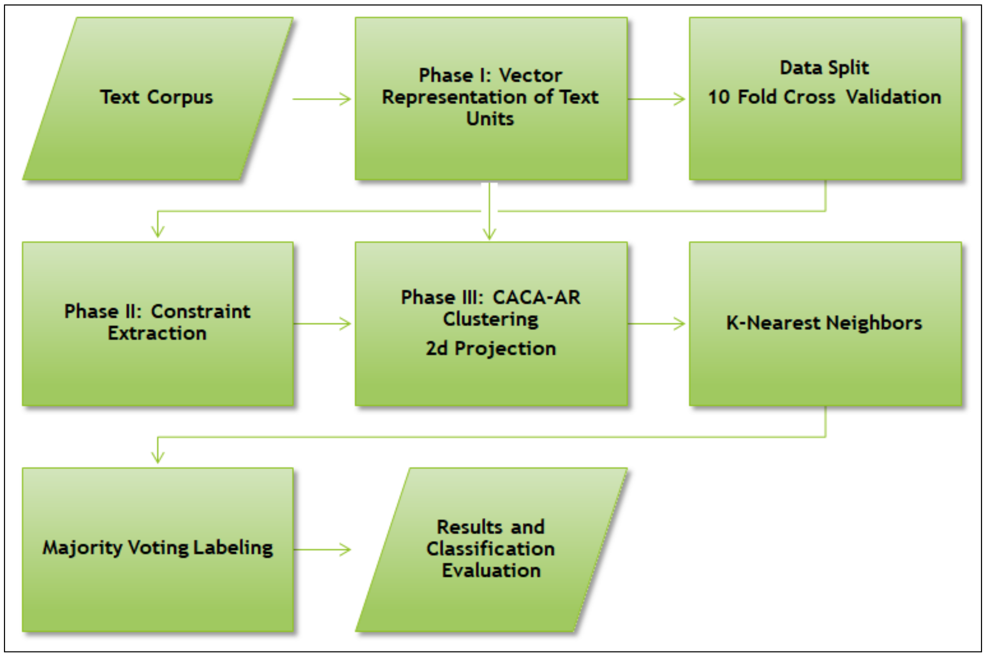

Text Corpus
Phase I: Vector Representation of Text Units
Data Split
10 Fold Cross Validation
Phase II: Constraint Extraction
Phase III: CACA-AR Clustering
2d Projection
K-Nearest Neighbors
Majority Voting Labeling
Results and Classification Evaluation

Figure 5.1: Clustering Framework for Aspect Identification and Sentiment Prediction

The clustering framework starts with a corpus of product reviews as an input and ends with labeled sentences. In the task of aspect category identification, the goal is to assign input sentences with aspect categories (e.g., food, price, ...,etc.), while the goal in sentiment prediction task is to assign input sentences with sentiment classes (e.g., positive, negative and neutral). In the next two sections, I describe the major phases in the framework for achieve each task.

5.1 Phase I: Vector Representation of Text Units

Both tasks, sentiment prediction and aspect category identification, require representing semantic of variable-length word sequences, such as phrases, constituents, sentences, paragraph or even entire document to be represented as fixed-length vectors in a high-dimensional space. Such representation aims at finding real-valued vectors that captures the semantic of word sequences whereby all semantically-related word sequences get to be closer in the space. Different models have been proposed to achieve this task. One of the earliest models, for instance, is the bag-of-words or bag-of-n-grams model (Harris [1954]). In this model, all unique words or n-grams are extracted from the given corpus to represent a vocabulary set or bag. Each sentence is represented as a vector of length that is equal to the vocabulary size. Given a sentence or a word sequence, most of the vector entries are zeros except for those entries corresponding to sentence words that are set to the frequency of the word. Although bag-of-words has been successfully used in many tasks of natural language processing, due to its simplicity and efficiency, it is not appropriate for sentiment prediction task. This is because bag-of-words loses word order in the representation and hence cannot capture adequate information about semantics of words. Moreover, it suffers from high data sparsity and dimensionality.

As a result of the success of word vectors (Mikolov et al. [2013a]), several attempts have been proposed to produce semantic vectors of word sequences using word vectors. The underlying intuition behind using word vectors is the formulation of distributional hypothesis, “words that occur in similar contexts tend to have similar meaning.” A direct result of the distributional hypothesis is that the meaning of a word is characterized by the context where it usually occurs (Turney and Pantel [2010]). Several distributional semantic models have been proposed to obtain vector representations for words. One of the earliest models, for instance, is the Hyperspace Analogue to Language (HAL) (Lund and Burgess [1996]). HAL model constructs semantic space based on co-occurrence matrix of words. HAL is extended to Correlated Occurrence Analogue to Lexical Semantics (COALS) by Rohde et al. (Rohde et al. [2006]). COALS combines latent semantic analysis and HAL model by employing the Pearsons correlation and singular value decomposition. In (Pennington et al. [2014]), Global Vectors (GloVe) model is presented. GloVe encodes the meaning of words by computing the ratios of word-word co-occurrence probabilities. In (Mikolov et al. [2013a]) and (Mikolov et al. [2013b]), two neural network for learning word vectors are introduced, Continuous Bag of Words (CBOW) and skip-gram. Both models are known as "word2vec". The former learns to predict the word given its context, while the latter learns to predict the context given a word. According to Mikolov et al., CBOW is several times faster to train than the skip-gram and gives slightly better accuracy for the frequent words, whereas skip-gram works better with small amount of the training data and represents well for even rare words or phrases. To sum up, distributional semantics models enable clustering method for aspect-based sentiment analysis tasks because they quantify the semantic similarity between words using cosine measure.

One method to represent a sentence as a vector, for example, is to construct semantic vector of a sentence is to compute a weighted average of all the vectors of sentence words (Mikolov et al. [2013b]). However, this model suffers from losing word order just like bag-of-word model. Another method suggests combining word vectors with respect to their order in syntactic tree using matrix-vector operations (Socher et al. [2010]). This second method, combining word vectors according to the sentence parse tree, can only produce vectors for sentences since it depends on the syntactic tree.

Recently, Le and Mikolov proposed paragraph vector model (Le and Mikolov [2014]), which is commonly known as "Doc2Vec". Doc2vec is an extension of word2vec model that learns distributed vector representations of variable-length word sequences which can be phrases, sentence or even entire document. There are two approaches in doc2vec model: Distributed Bag of Words (DBOW) and Distributed Memory Paragraph Vector (DMPV). As explained by Lau and Baldwin [2016], DBOW model is like the word2vec skip-model because it ignores context words in the input, and it forces the model to words that are randomly sampled from the output. DMPV, by contrast, works in a similar way to word2vec CBOW. DMPV introduces a special token for a document in addition to multiple target words. Unlike CBOW, word vectors are not summed but concatenated.

In this thesis, I adopt paragraph vector model to create distributed semantic vectors of sentences for both tasks of ABSA. A major advantage of paragraph vectors is that they can be learned from unlabeled data; hence, they can work well in domains that do not have enough labeled data, such as ABSA. In addition, paragraph vector model preserves semantic of words because it takes order of words into consideration.

5.2 Phase II: Pairwise Constraint Extraction

The goal of this phase is to automatically generate sets of must-linked and cannot-linked constraints (discussed in section 2.1.2) for aspect category identification and sentiment prediction. There are two approaches to achieve this end. The first one is to ask domain experts (linguists) to specify whether and when two sentences have the same sentiment or refer to the same aspect category. Another approach is to generate pairwise constraints from a given labeled dataset. For instance, given a labeled datasets for ABSA (Pontiki et al. [2014, 2015]). To generate k constraints, we select k pairs of sentences iteratively. When both sentences have the same label (i.e., both belong to the same aspect category), the pair is added to the set must-linked constraints. Otherwise, the pair is added to the set of cannot-linked constraints. For the task of aspect category identification, for instance, we use the category label to generate constants. Similarly, we use sentiment label to generate constraints for the task of sentiment prediction.

Both sets of constraints can be further expanded by applying the transitive closure on must-linked constraint set and entailment property on cannot-linked constraint set as described in section 2.1.1. A simple algorithm for this approach is described in Algorithm 3. However, as I mentioned previously in section 2.1.2, random selection of constraints is ineffective because the relation of some selected pairs of data can be trivially determined by a clustering algorithm. A more informative way is to select pairs of sentences where both sentences have the same label and the semantic similarity between their semantic vectors is low, or pairs of sentences where both sentences have different labels and the semantic similarity between their vectors is high.

5.3 Phase III: Apply ACA with Pairwise Constraints

The goal of this phase is to group sentence vectors (output of phase I) given the extracted pairwise constraints (output of Phase II) into disjoint clusters where

Algorithm 3 Pairwise Constraint Generation

Input: labeled dataset $\{x_i, y_i\}_{i=1}^{n}$, number of constraints k

Output: Must-Linked (ML) set, Cannot-Linked (CL) set

```
1:  $ML \leftarrow \phi$, $CL \leftarrow \phi$
2:  for $i = 1$ to $k$ do
3:      select two instances $x_i, x_j$ randomly
4:      if $y_i = y_j$ then
5:          $ML \leftarrow ML \cup (x_i, x_j)$
6:      else
7:          $CL \leftarrow CL \cup (x_i, x_j)$
8:      end if
9:  end for
10: expand ML set using its transitive closure
11: expand CL set using CL entailment
```

each cluster represents a collection of sentences that hold the sentiment category. Similarly, the outputs from phase I and phase II serve as input in phase III for aspect category identification. The main task of phase III is to carry out the proposed ACA with pairwise constraints to group sentence vectors into disjoint clusters where each cluster represents aspect category in the domain.

5.4 Cross-Validation for Constrained Clustering

Constrained clustering can be evaluated using K-Fold Cross Validation technique as described in Pourrajabi et al. (Pourrajabi et al. [2014]).

The results of CACA-AR are calculated using 10-fold cross-validation technique as shown in Figure 5.2. In this technique, the labeled data is partitioned into 10 folds such that 9 folds form a training data, and 1 fold is held out as a test data for evaluation. The training data is used to derive ML and CL constraints. The entire unlabeled data (10 folds without labels) along with the generated constraints and initial parameters of CACA-AR are fed to the constrained clustering algorithm (CACA-AR) to find the data clusters. The k-Nearest Neighbor (KNN labeling) assigns a label for each instance in test fold by applying a majority-voting technique using nearest neighbors of the instance. Once the test fold is labeled, the accuracy metrics (*recall, precision, and F-score*) for the test fold can be computed with respect to the actual labels, as given by the ground truth. The entire technique is then repeated 10 times so that each fold is used as a test fold and as a training fold.

The accuracy measure of the algorithm reported by 10-fold cross-validation is then the average of the values computed for each fold. This technique can be computationally expensive but does not waste too much data, which is a significant advantage when the size of labeled data is small.

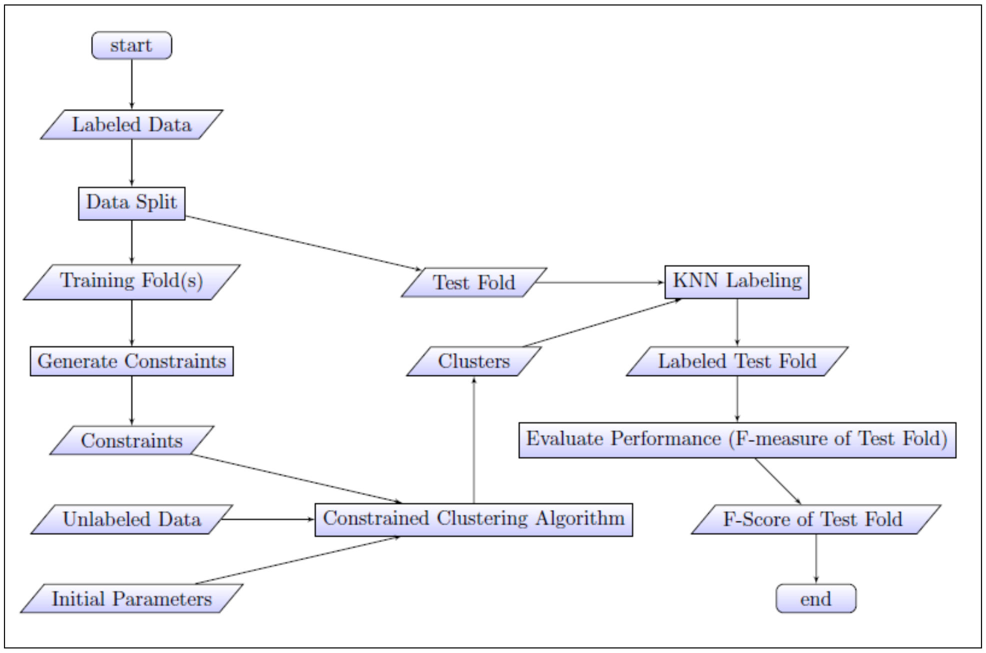


Figure 5.2: A single Step in an N-Fold Cross Validation for Constrained Clustering

5.5 ACA Data Labeling Using k-Nearest Neighbor

ACA-AR with constraints can be used as neighbors-based classification approach. As shown in Figure 5.3, ACA-AR finds clusters within each data class by making nearest neighbors for each data instance adjacent to that instance on a 2d-grid. Therefore, the algorithm can be used as neighbor-based classification approach. For instance, given a data instance d at grid location (x, y) and a fixed radius r, the neighbors of d are the set of instances $\{d_i\}$ such that the grid location for each d_i is within the area $(x \pm r, y \pm r)$. Classification is then computed from a majority vote of the nearest neighbors for each test instance; a test instance is assigned the data class which has the most representatives within the nearest neighbors of the instance.

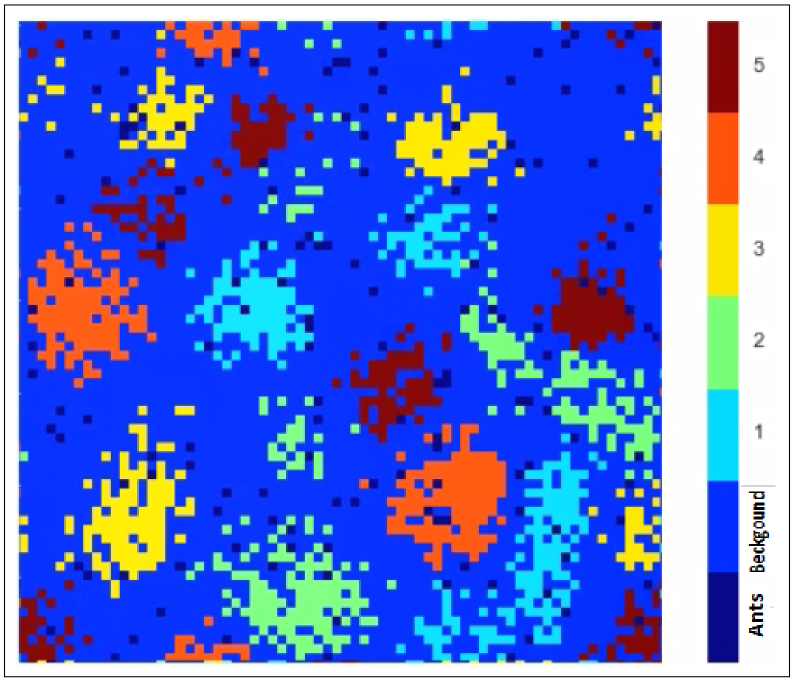


Figure 5.3: Constrained Clustering using CACA-AR a dataset of 1500x20 real-valued vectors. The dataset consists of 5 classes color coded with class labels where each class is comprised of 3 clusters. The algorithm finds data clusters within each class

Chapter 6

Evaluations: Results and Discussion

This chapter is organized as follows: the details of adopted benchmark datasets for evaluation are presented in section 6.1. In section 6.3, I explain how vector representation of text units (sentences and words) is achieved. For result comparison, I adopt lexicon-based approach (unsupervised) and logistic regression classification (fully supervised) because they are the most frequently used approaches for sentiment analysis. Both baselines are presented in section 6.4. In section 6.2, the evaluation measures, recall, precision and F1-score are defined. I explain the implementation of ACA-AR as well as the feasibility of its parallel counterpart in section 6.5. Finally, I present the results of CACA-AR in sections 6.7 and 6.8.

6.1 Benchmark Datasets

In this section, I highlight the benchmark datasets used in this thesis.

- SemEval Dataset (Pontiki et al. [2014], Pontiki et al. [2015]): SemEVal Task 12 (the International Workshop on Semantic Evaluation). This dataset was introduced to standardize the evaluation process of ABSA. The dataset is comprised of product reviews that cover three domains: restaurants, laptops and hotels. The total number of reviews and sentences for each domain is shown in Table 6.1. The laptop and restaurant reviews are manually labeled at the sentence-level taking into account the context of the review. Each sentence is primarily assigned two labels: the target of sentiment and the sentiment label (polarity). Since this dataset is dedicated to ABSA, the first label, target of sentiment, is expressed as a pair of an entity label and an attribute label. For the laptop domain, for example, there are 22 predefined entity labels E (e.g. laptop, display, CPU, motherboard, hard disk, memory, battery, etc.) and 9 predefined attribute labels A (e.g. general, price, quality, performance, etc.). The second label, sentiment label S, is the polarity of the sentiment that is expressed towards the target. The polarity set includes $\{positive, negative, neutral\}$. Each E#A pair is assigned to S. Consider, for example, the following sentences with their corresponding labels (Pontiki et al. [2015]):
  - S1: *It is the worst laptop ever.*→ {laptop#general, negative}
  - S2: *The applications are also very easy to find and maneuver.* → {software#usability, positive}

- S3: *Sometimes you will be moving your finger and the pointer will not even move.* → {mouse#operation_performance, negative}

Similarly, each sentence in the restaurant reviews is labeled with the target and sentiment polarity. The target label involves 6 entity labels (restaurant, food, drinks, service, ambiance, location), and 5 attribute labels (general, price, quality, style, miscellaneous), which results in 12 possible combinations. Consider, for example, the following sentences:

- S1: *I was very disappointed with this restaurant.* → {restaurant#general, negative}
- S2: *Food was okay, nothing great.* → {food#quality, neutral}
- S3: *The fajitas were pretty expensive.* → {food#price, negative}

For this dataset, I use CACA-AR to evaluate the accuracy of sentiment prediction and aspect category identification (target) for laptop and restaurant sentences.

Table 6.1: SemEval (Task 12) Dataset

	Laptops	Restaurants	Hotels
	Training Data		
Reviews	277	254	
Sentences	1429	1183	
	Test Data		
Reviews	173	96	30
Sentences	761	685	266

- Stanford Sentiment Treebank (SSTB) (Socher et al. [2013]): This dataset is comprised of 10,752 single sentences extracted from movie reviews, which was introduced in (Pang and Lee [2008]). Each sentence is parsed into phrases using Stanford parser. The dataset provides the polarity for each sentence as well as the polarity for each phrase within the sentence. The total number of labeled phrases is 215,154. Unlike SemEval dataset, each sentence is assigned a sentiment category (polarity) out of five categories: very negative, negative, neutral, positive and very positive. The number of the sentences for each category is shown in Table 6.2. For this dataset, I evaluate CACA-AR accuracy for sentiment prediction at the sentence level.

The distributions of sentiment classes for all datasets are shown in Table 6.2

Table 6.2: Class Distribution for Sentiment Datasets

class	SSTB	Laptops	Restaurants
Very Negative	1370	-	-
Negative	2850	882	593
Neutral	2013	170	92
Positive	2832	1138	1156
Very Positive	1687	-	-
Total	10752	2190	1841

6.2 Accuracy Measures

Since the tasks of ABSA are considered multi-class classification tasks, a wide diversity of measures can be used to evaluate the accuracy of sentiment analysis models. The simplest measure, for example, is to evaluate performance by computing the plain accuracy which is the percentage of correctly predicted class labels over all predictions. However, 'high' accuracy alone is deceptive in many cases. For example, assume that we have a Spam detection system that predicts if an email is a Spam or not. However, the system is always faulty, i.e., it always predict that any email is not a Spam. Given a test data of a hundred emails where only one email is a Spam and 99 are non-Spam's. The accuracy of the system in this case is 99% as the prediction is correct for non-Spam emails. While, in fact, the system does nothing as it fails to predict the class (Spam), which is actually the goal of the system. To overcome this issue in predictive analytics, the accuracy of the classification systems is reported by *recall, precision* and *F1-score* measures.

6.2.1 Illustrative Example: Recall, Precision and F1-score

Consider a classification system that can classify (predict) whether a sentiment polarity expressed in a given sentence is positive, negative or neutral. Suppose that we are given a test data that comprised of manually-labeled sentences (ground truth) where each sentence is labeled with one sentiment polarity (positive, negative or neutral). The test data involves 220 sentences: 100 negative sentences, 100 positive sentences and 20 neutral sentences. The 220 sentences are fed to the classification system to get a predicted polarity for each sentence. A clean and obvious way to present the prediction/classification results is to use a *confusion matrix*, also known as contingency table, as illustrated in Table 6.3.

Table 6.3: Confusion Matrix

		Predicted			
		negative	neutral	positive	total
Actual	negative	40	10	50	100
	neutral	7	5	8	20
	positive	30	25	45	100
	total	77	40	103	220

Since we have three sentiment classes (positive, negative and neutral), the confusion matrix has 3 rows and 3 columns, disregarding the total column. The rows (*actual*) represent how the classification system predicted each class. For instance, out of the 100 actual negative sentences (first row), 40 sentences were *correctly* predicted as negative, 10 sentences were *incorrectly* predicted as neutral, and 50 sentences were *incorrectly* predicted as positive. Similarly, out of the 100 actual positive sentences (third row), 45 sentences were *correctly* predicted as positive, 25 sentences were *incorrectly* predicted as neutral, and 30 sentences were *incorrectly* predicted as negative. Therefore, we can see from the matrix that the system in question has trouble distinguishing between positive and negative sentences. This is because almost half of the actual negative sentences were predicted as positive, and half of the actual negative sentences were predicted as positive. The matrix also shows that the classifier has a clear weakness in predicting the neutral class. This is because only 5 sentences out of 20 actual neutral sentences were correctly predicted.

All correct predictions are located in the left diagonal of the matrix, while the errors (false predictions) are represented by values outside the diagonal. The plain accuracy is, therefore, $(40 + 5 + 45)/220 = 0.41$. In case of a perfect prediction, the diagonal entries would be 100, 20, 100, respectively, while the rest of the entries would be zeros. The confusion matrix can be unnormalized, such as the one shown in Table 6.3, or normalized. The normalization is achieved by dividing each entry in the unnormalized confusion matrix by the total number of instances (sentences), given in the last column in Table 6.3. The normalized form of Table 6.3 is shown in Table 6.4.

Table 6.4: Normalized Confusion Matrix

		Predicted			
		negative	neutral	positive	support
Actual	negative	0.4	0.1	0.5	100
	neutral	0.35	0.25	0.4	20
	positive	0.3	0.25	0.45	100

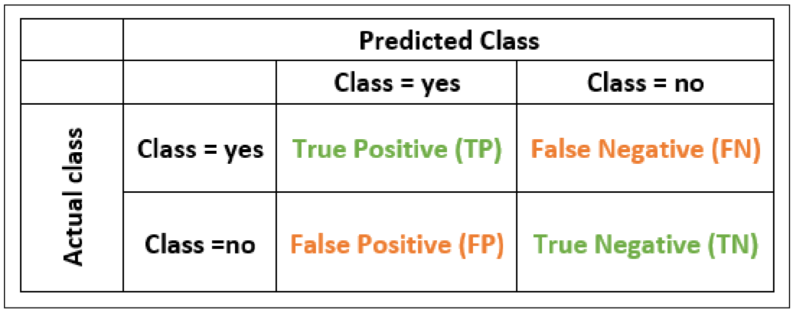
Given a confusion matrix, a more in-depth evaluation of the classification system is to compute the average *recall*, *precision* and *F1-score* for the classes. To achieve this, however, we further summarize the results by building a 2x2 confusion matrix for *each observed class* as shown in Figure 6.1. For each class, we count the following four entries:

(i) True Positive (TP) is the value when the predicted class is "yes" and the actual class, as given by the ground truth, is "yes".

(ii) False Positive (FP) is the value when the predicted class is "yes", and the actual class is "no".

(iii) True Negative (TN) is the value when the predicted class is "no", and the actual class is also "no".

(iv) False Negative (FN) is the value when the actual class is "yes", and the predicted class in "no".



		Predicted Class	
		Class = yes	Class = no
Actual class	Class = yes	True Positive (TP)	False Negative (FN)
	Class =no	False Positive (FP)	True Negative (TN)

Figure 6.1: Confusion Matrix for Particular Class, green cells: correct predictions, red cells: incorrect predictions

Consider, for instance, the *negative class*. Based on the values given in Table 6.3, the *True Positive* for this class is 40, which is the value where the actual class is "negative", and the predicted class is also "negative". Similarly, the *True Negative* is the the value wherein the actual class is "non-negative" (i.e., either "neutral" or "positive"), and the predicted class is also "non-negative" (i.e., either predicted "neutral" or "positive"). Therefore, *True Negative* for the "negative" class is $(5+8+25+45 = 83)$. The *False Negative* is the value wherein the actual class is "negative" and the predicted class is "non-negative", which is $(10+50)$. By contrast, the *False*

Positive is the the value wherein the actual class is “non-negative” but the prediction class is “negative”, that is $(7 + 30)$. The computation for TP, TN, FP and FN for the three classes are shown in Table 6.5

Table 6.5: TP, TN, FP and FN Computations

	True Positive	**True Negative**	**False Positive**	**False Negative**
Negative	40	$(5+8+25+45) = 83$	$(7+30) = 37$	$(10+50) = 60$
Neutral	5	$(40+50+30+45) = 165$	$(10+25) = 35$	$(7+8) = 15$
Positive	45	$(40+10+7+5) = 62$	$(50+8) = 58$	$(30+25) = 55$

Given the values listed in Table 6.5, the recall, precision and F1-score can be computed for each class as follows:

- $recall = \frac{TP}{TP+FN}$
- $precision = \frac{TP}{TP+FP}$
- $F1 - score = \frac{2 \times recall \times precision}{recall + precision}$

After substituting the values listed in Table 6.5 in the above equations, we get the values for recall, precision and F1-score as shown in Table 6.6

Table 6.6: Recall, Precision and F1-score

	Recall	**Precision**	**F1-score**
Negative	0.4	0.52	0.45
Neutral	0.25	0.13	0.17
Positive	0.45	0.44	0.44
Average	0.37	0.36	0.35

6.2.2 Interpretation of Recall, Precision and F1-score

This section provides the intuition as well as the interpretation of using recall, precision and F1-score to report the accuracy of classification systems. In addition, it provides the interpretation of using these measures to evaluate the accuracy of sentiment analysis systems.

- recall: given all instances that should have *the same class label* (e.g. positive sentences), *recall* measures how many of these were correctly predicted. In the context of sentiment analysis, *recall* indicates the ability of the system to predict a certain class. For instance, *recall* value of neutral class determines how accurately the system recognizes neutrality.

- precision: given all the *predicted labels* for a certain class, precision indicates how many of the instances were correctly predicted. For sentiment analysis, *precision* indicates how often the predicted sentiment class are correct.

- F1-score: is the harmonic mean of *recall* and *precision.* The range of F-score is $[0, 1]$ with 1 being the perfect score. Reporting accuracy in terms of F1-score provides a single value that equally rates the significance of *recall* and *precision* together. This measure is also known as F-score or F-measure.

For the results shown in Table 6.5, we can observe, for example, that the system suffers from predicting the neutral class as only 40% of actual neutral sentences were hit by the classifier. Besides, the precision (predictive rate) of the classifier to determine neutral sentences is 13%. The overall performance of the classifier can be given as the average of recall, precision and F1-score. The overall accuracy of the classification system, explained in section 6.2.1, is 34% while the plain accuracy is 41%.

6.3 Feature Vectors of Sentences

In this thesis, I adopt the paragraph vector model (Le and Mikolov [2014]) to create distributed semantic vectors of sentences (real-valued feature vectors). A major advantage of paragraph vectors is that they can be learned from unlabeled data; hence, they can work well in domains that do not have enough labeled data, such as ABSA. In addition, paragraph vector model preserves semantic of words because it takes the order of words into consideration. To generate the vectors, I train a paragraph vector model using "Doc2Vec", a Python implementation of the paragraph vector model. The training data consists of 665,276 sentences as shown in Table 6.7.

Table 6.7: Training Data

Dataset	Sentences
SemEval-Laptops	2,190
SemEval-Restaurants	4,031
Tackstroom and McDonald (TM)	6,546
Stanford Sentiment Treebank (SSTB)	17,298
Amazon Reviews	635,211
Total	665,276

As recommended by (Řehůřek and Sojka [2010]; Lau and Baldwin [2016]), I use the following parameters to train the model:

- training algorithm: distributed memory (Le and Mikolov [2014]) [1]
- dimensionality of the feature vectors: 300
- window (maximum distance between the predicted word and context words used for prediction within a sentence): 10
- minimum word frequency: 10

The training algorithm iterates for 20 epoch with shuffled sentences in each iteration. After completing model training, each sentence is represented by a real-valued vector of dimension 300.

6.4 Baselines

I compare the results of constrained clustering based on CACA-AR against two approaches: unsupervised classification using lexicon-based and fully supervised classification using logistic regression classifier.

6.4.1 Lexicon-based Approach

As I previously explained in section 4.6.2, lexicons are common lexical resources for sentiment analysis. As a first experiment, I use lexicon-based sentiment analyzer using SentiWordNet (Esuli and Sebastiani [2007]) with vote-flip algorithm (Choi and Cardie [2009]) to evaluate sentiment polarity of sentences. SentiWordNet provides positive,

[1]Neural-network language model to learn distributed representations (fixed-length real-valued feature vectors) for variable-length word sequences

negative and objective (neutral) sentiment scores in a continuous scale $(0.0, 1.0)$ for a given word based on its part-of-speech (noun, verb, adjective, adverb).

For example, querying SentiWordNet for the verb *dislike* gives $(PosScore = 0.0, NegScore = 0.5)$, which indicates that *dislike* conveys a negative sentiment. If word is objective (neutral), SentiWordNet returns one for objectivity score and zero for positive and negative scores. To classify sentences, sentence is first tokenized into words, and each word is tagged with its part-of-speech using part-of-speech tagger. Next, the sentiment scores is retrieved for each pair (word, tag) in the sentence so that words are classified into positive, negative and neutral. Finally, the sentiment polarity of words are fed into the *vote-flip* algorithm to evaluate the sentiment polarity of a sentence. vote-flip algorithm is a rule-based algorithm that uses the counts of positive, negative and neutral words as well as the existence of negation to determine the sentiment polarity of a sentence.

As explained previously in section 4.6.2, lexicon-based approach to sentiment analysis depends on retrieving sentimental information about words from a pre-compiled lexicon. Sentimental information includes word polarity, sentiment intensity and objectivity. The lexicons can be automatically or manually compiled. The compilation of lexicons is still an active line of research in the sentiment analysis community. However, a major challenge in this approach is that lexicons do not provide information about the contextual polarity [2] of words (Wiebe et al. [2005]). Since the sentiment is highly context-dependent (i.e., the prior polarity of a word or a phrase is different from its contextual polarity). This problem significantly affects the accuracy lexicon-

[2]The contextual polarity of a given word is the polarity of the word when it is used in a particular context

based sentiment analysis. Consider, for example, the scores assigned to the adjective *long* as given by Sentiwordnet lexicon: Positive: 0.25, Negative: 0.125 and Objective: 0.625. The scores indicate that *long* tends to be objective (i.e., neutral). However, consider the impact of context when *long* is used in sentences such as: *The movie was too long*, *battery life typically does not last long*, or *battery can last a very long time*. The polarity of the three sentences will be considered neutral by lexicon-based approach. While, in fact, the polarity of the first sentence,*The movie was too long*, is negative. the sentence, *battery life typically does not last long* holds a negative sentiment. By contrast, the third sentence, *battery can last a very long time*, indicates a positive sentiment.

6.4.2 Multi-class Logistic Regression

Logistic regression model is one of the most frequently used model for data classification tasks. Unlike other classification methods, such as support vector machines, decision trees or k-nearest neighbor, logistic regression expresses the probability of outcome as a liner predictor function $f(x, \theta)$ such that, $p(y|x) = f(x, \theta)$. θ is a vector of parameters which are usually estimated by maximum likelihood technique for a given dataset. In logistic regression, $f(x, \theta)$ is known as a parametric method.

Generally, logistic regression predicts the probability of occurrence of an event by fitting data to a logistic function. For instance, if outcome variable y is binary (0, 1), logistic function is given by: $p(1|x, \theta) = \frac{1}{1+\exp(-z)}$ and $p(0||x, \theta) = 1 - p(1|x, \theta)$ here, z is a linear function of the predictor variables such that $z = \beta_0 + \beta_1 x_1 + ...\beta_n x_n$, where β_0 is constant, and $\beta_{1,n}$ are predictor variable coefficients or regression coefficients and $x_{1,n}$ are predictor variable values. This logistic transformation forces probability estimates to be between 0 and 1 regardless of the value of z. The predictor variable coefficients are computed using a maximum-likelihood technique. If the estimated probability of the event under consideration is less than 0.5, then it is concluded that the event will not occur. In contrast, if the estimated probability of the event is greater than 0.5, it is inferred that the event will occur. If, however, the estimated probability exactly equals 0.5 then no inference concerning the occurrence of the event can be made.

Multi-class logistic regression is an extension of binary logistic regression. There are many reasons to use multi-class logistic regression for sentiment classification. First, it is applicable when outcome variable is to be classified to one of multiple possible classes. Second, unlike liner regression model or general linear regression model, logistic regression can be used whether independent variables are statistically independent or not. Third, independent variables can be discrete or continuous. Fourth, it can handle non-linear relationships between dependent variable and independent variables. Lastly, no assumptions regarding linearity, normality or homoscedasticity [3] are required.

To generate the results of logistic regression, I train one-versus-rest multi-class logistic regression classifier using (scikit-learn developers) with 10-fold cross validation with the following parameters:

- inverse of regularization strength, c,range: $[1 \times 10^{-10}, 1 \times 10^{10}]$.
- penalty: L2

[3]The relationship between the independent variable and the dependent variables is the same across all values of the independent variables

- tolerance: 1×10^{-6}

6.5 Implementation and Experiments

In this section, I present the ACA implementation and its parallel counterpart.

6.5.1 ACA Simulator

In order to evaluate the results of ACA-AR, I built a simulator that visualizes ACA throughout the clustering process using Python 3.6. The core of the simulator includes ACA-AR and CACA-AR implementation. Both algorithms are programmed as a multi-threaded object-oriented application. The implementation consists of three main classes:

- Grid class: is basically a 2D array of data objects. Grid class provides two main methods. The first method scatters data objects and ants across grid cells. The second method computes kernel density estimation given a certain area on the grid. The grid class also provides other utilities for transforming grid contents into images for visualization purposes.

- Data class: this class manipulates data objects. Each datum object consists of real-valued vector X, actual label Y and predicted label. Each datum is initialized with a unique identifier (ID). Data class provides methods for computing data similarity (Cosine measure, Euclidean distance, ...etc).

- Ant class: this class provides the specification of ants. Each ant is actually an object that runs a thread method. Ant class specifies ant movement, pick-up and drop-off methods.

In summary, the simulator monitors clustering process for each ant step.

6.6 Parallel Implementation

The algorithm is implemented on the multi-core GPU and CPU machines using CUDA and OpenMP, respectively.

On the GPU, the algorithm starts from the CPU. The CPU reads in the data file and creates two arrays: ants and objects, which store their locations on the grid. The grid is created as an array (integer) initialized to -1. Then, the ants and objects are randomly placed on the grid. The arrays and grid are moved from the CPU to the GPU global memory to avoid communication latency between the CPU and GPU. The CPU also calculates the distances between each objects using the Euclidean distance, which is also passed to the GPU. Algorithm 1 is executed on the GPU. Each ant is a block in the GPU. We restricted each block to one thread since there was no performance gain when more ants were added, creating unnecessary synchronization latency and race conditions. The computations of each ant, a_i, is very fine-grained. There are three operations performed by each ant: pick-up, drop-off and move.

Each ant can be categorized as either loaded or not loaded. If the ant is not loaded, the ant considers its current location: (i) if empty it moves randomly to the adjacent cells; (ii) otherwise, the ant applies Equation 3.5 to determine whether to pick up the object. The number computed by Equation 3.5 is compared to a random number ($rand$) between 0 and 1 generated by the ant. If the pick-up probability determined by the Equation is larger than $rand$, the ant picks up the object and continues to wander, otherwise, it moves to another grid location. On the other hand, if the ant is loaded, the ant calculates the next drop-off probability using Equation 3.6. Again, a random number is generated and compared to the drop-off probability. If the drop-off probability is greater than the generated random number the ant drops the object. This implies that the objects are all similar. Otherwise, the ant moves to another adjacent cell. Note that, only one ant may be located in each grid location.

The termination condition is the total number of iterations.The ants radius of perception is increased to cover up to 1/4th of the grid area. Each ant starts with radius one. This results in forming a large number of small dense clusters because ants become unable to perform pick-ups. To merge the small clusters into larger dense ones, the ant gradually increases its radius of perception to recognize more dissimilarity, which, in turn stimulates pick-ups. When the ant covers up to 1/4th of grid area, it gradually decreases its radius to compact the large clusters. The final result is sent back from the GPU to the CPU. The algorithm is very similar on the CPU, with the exception of sending data between CPU and GPU.

6.6.1 The Feasibility of Parallel Implementation

Although all ants wander simultaneously on the grid, only those ants with non-overlapped neighborhoods can compute a pick-up or drop-off probability at the same time. This is because each ant considers its neighborhood as a critical section when it computes a pick-up or a drop-off probability. To highlight the impact of such synchronization mechanism on the performance, I (Qasem et al. [2017]), report speedups of both parallel implementation strategies by varying the radius of perception from 1 to 6. In all experiments, we take the number of ants (threads) as 1% of data size N. Figure 6.2 shows speedup results for OpenMP implementation. Speedup increases with respect to the data size for small values of radius of perception ($r = 1, 2$). This result is due to the fact that larger number of threads can be executed simultaneously. However, speedup decreases rapidly as the radius increases because of race condition. In addition, the increase in radius results in collecting a larger data sample within ant's neighborhood. This,in turn, makes computing KDE, pick-up and drop-off more computationally-intensive.

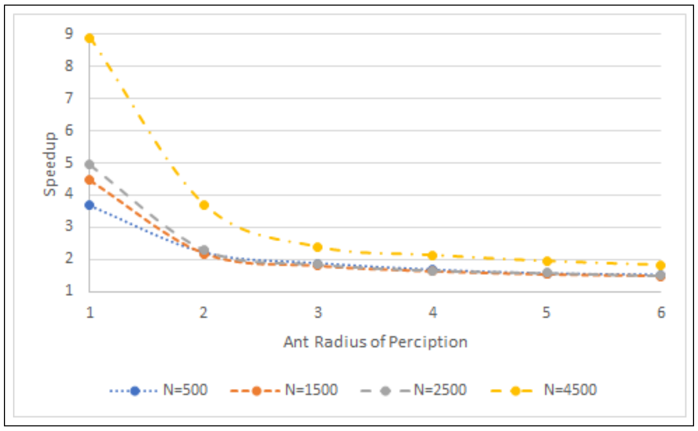


Figure 6.2: OpenMP-based Results

Unlike OpenMP results, The gained speedup on GPU implementation increases up to 39x compared to the sequential implementation as shown in Figure 6.3. Since the most expensive computation in our algorithm is due to the KDE, we consider a parallelization strategy that offload KDE computation by each ant to block-level at GPU. In GPU implementation, each ant is mapped to a block so that independent ants can run at the same time. Since the major task of each ant is to compute KDE using the objects within its neighborhood, we partition ant task into finer subtasks that handle KDE computation (i.e., summation of kernels of probability density function for an object x_i).

Notice that a significant speedup can be gained when KDE is implemented in a data-level parallel manner in each GPU block. This is because sequential KDE computation for n data objects of dimension d requires $O(n^2d)$ (Michailidis and Margaritis [2013]). In addition to that, parallel implementation of multivariate KDE can entirely get the benefit of data-level parallelism because it is embarrassingly parallel (straightforward vectorization). This parallel strategy reduces the impact of ant synchronization mechanism, especially for ants with large radius of perception.

To reduce latency of global memory access, each ant transfers the data object to be picked-up or dropped-off to the register level, and the rest of the objects within its neighborhood to the shared memory level. This significantly reduces global memory accesses because such objects are the most frequently accessed data object in KDE computation. This strategy enables a scalable GPU-based implementation with an increase in speedup up to 39%

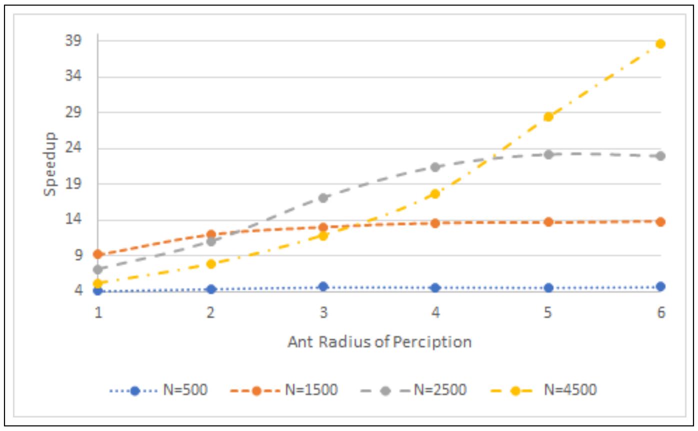


Figure 6.3: GPU based Results

6.7 Results of Sentiment Prediction

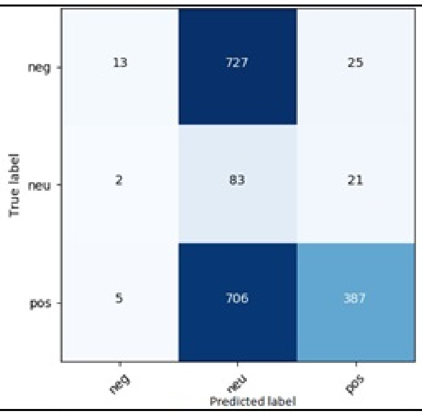
This section presents the results of sentiment prediction task for ABSA. For the three approaches: Lexicon-based, Logistic regression and CACA-AR. I report the recall, precision and F1-score for each dataset as well as the confusion matrices.

6.7.1 Lexicon-based Approach

The confusion matrices and the scores of recall, precision and F1-score for Lexicon-based are shown in Figure 6.4 for each dataset. According to the confusion matrices, lexicon-based suffers from identifying the negative and the positive classes from the neutral class in the three datasets. For the laptop dataset, for example, only 13 sentences out of 765 negative sentences were correctly predicted as negative while 727 sentences were incorrectly predicted as neutral. Similarly, out of the 1098 positive sentences, only 387 were correctly predicted as positive while 706 were incorrectly predicted as neutral. This same weakness can be seen in restaurant and SSTB datasets. This weakness significantly lowers recall values of negative class (0.02) and positive class (0.35). Another direct impact of this weakness is the low precision of neutral class (0.05). Although lexicon-based correctly identified most of neutral sentences (recall 0.78), its precision is very low (0.05). This is because 1433 sentences were incorrectly predicted as neutral (i.e., false positive of neutral class is 1433). In conclusion, the results demonstrates that lexicon-based approach does not capture contextual polarity of words. In addition, notice that the average F1-score varies from 0.26 for SSTB (sentences of movie reviews) to 0.39 for restaurant dataset as a result of domain dependency of words.

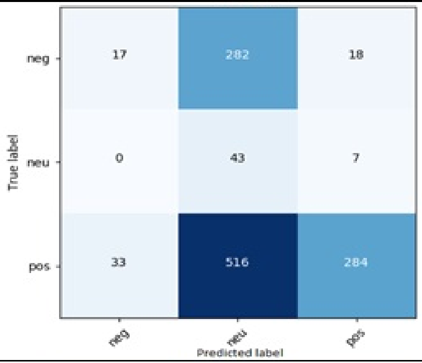
Laptop dataset

	precision	recall	f1-score	support
neg	0.65	0.02	0.03	765
neu	0.05	0.78	0.10	106
pos	0.89	0.35	0.51	1098
avg / total	0.75	0.25	0.30	1969



Restaurant Dataset

	precision	recall	f1-score	support
neg	0.72	0.04	0.08	317
neu	0.05	0.86	0.10	50
pos	0.92	0.38	0.53	833
avg / total	0.83	0.31	0.39	1200



SSTB Dataset

class	precision	recall	f1-score	support
neg	0.73	0.04	0.07	4222
neu	0.20	0.87	0.32	2013
pos	0.71	0.28	0.40	4519
avg / total	0.62	0.29	0.26	10754

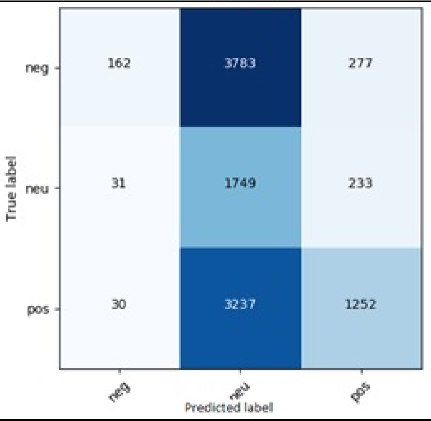


Figure 6.4: F1-score, Precision and Recall - Lexicon-based Approach: pos: positive class, neu: neutral class, neg: negative class. We can see that Lexicon-based performs poorly in identifying pos and neg sentences

6.7.2 Logistic Regression Approach

The confusion matrices, recall, precision and F1-score for multi-class logic regression (LR)[4] are shown in Figure 6.7 for each dataset. Unlike Lexicon-based approach, LR suffers from identifying neutral sentences in the three datasets as the classifier incorrectly predicts them as either positive or negative. For the laptop and restaurant datasets, for example, the true positive value of neutral class is zero. There are reasons for this. First, the classes are highly unbalanced and overlapped. For instance, the number of neutral sentences (support) in laptop and restaurant datasets is (105 and 50, respectively). Second, the feature vectors do not capture sufficient information about neutrality. Another weakness is that many negative sentences were incorrectly predicted as positive sentences. For the restaurant dataset, for instance, only 14 sentences out of 317 were correctly predicted as negatives while the rest were incorrectly predicted as positives. This results in a low recall value for negative class (0.04). (Pontiki et al. [2015]) report results of using different machine learning approaches and feature representations.

For the SSTB dataset[5], the sentiment classes are more fine-grained, very negative, negative, neutral, positive and very positive. As shown in the SSTB confusion matrix, most of the very negative sentences were incorrectly predicted as negative. Similarly, most of very positive sentences were incorrectly predicted as positive. This results in low recall values for very negative and very positive classes (0.00 and 0.15, respectively), and in lowering the precision of negative and positive classes (0.36 and 0.37).

[4]The used LR classifier id built by scikit-learn developers while feature vectors are generated using paragraph vector model as described in section 6.3

[5] SSTB: Stanford Sentiment Treebank, the details of this dataset is presented in section 6.1

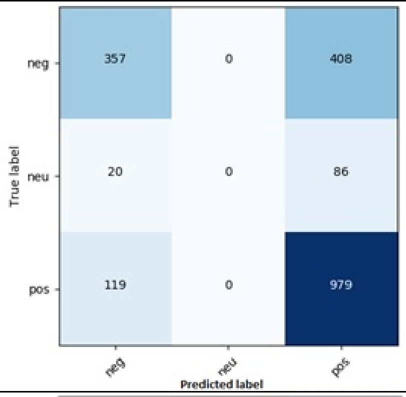
Predicting neutral sentences is also a weakness in this dataset. Most neutral sentences were incorrectly predicted as either positive or negative sentences. Therefore, the recall neutral class is (0.04).

For more state-of-the-art sentiment models, (Barnes et al. [2017]) evaluate sentiment prediction of this dataset using LSTM and Bi-LSTM[6] models using word embedding feature vectors (Mikolov et al. [2013b]). The results of such models demonstrates the same weakness in predicting very negative, very positive and neutral classes. The results of LR demonstrates that feature vectors of sentences, which are generated by paragraph vector model *do not* capture neutrality as well as sentiment information about fine-tuned classes, such as, very positive and very negative.

[6]LSTM: Long Short Term Memory Networks, Bidirectional-LSTM: are special kinds of Recurrent Neural Networks, capable of learning long-term dependencies

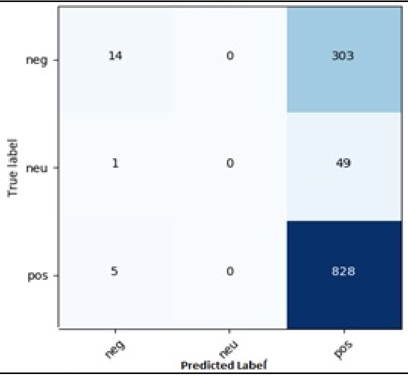
Laptop dataset

	precision	recall	f1-score	support
neg	0.72	0.47	0.57	765
neu	0.00	0.00	0.00	105
pos	0.66	0.89	0.76	1098
avg / total	0.65	0.68	0.64	1969



Restaurant Dataset

	precision	recall	f1-score	support
neg	0.70	0.04	0.08	317
neu	0.00	0.00	0.00	50
pos	0.70	0.99	0.82	833
avg / total	0.67	0.70	0.59	1200



SSTB Dataset

	precision	recall	f1-score	support
very neg	0.21	0.00	0.00	1371
neg	0.36	0.71	0.48	2851
neu	0.30	0.04	0.07	2013
pos	0.37	0.57	0.45	2832
very pos	0.46	0.15	0.23	1687
avg / total	0.35	0.37	0.29	10754

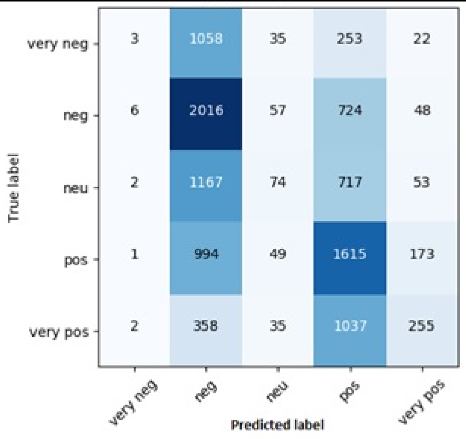


Figure 6.5: F1-score, Precision and Recall - Logistic Regression Approach: very neg: very negative class, neg: negative class, neu: neutral class, pos: positive, very pos: very positive

6.7.3 CACA-AR Approach

The results of CACA-AR are generated as described in sections (5.4;5.5). The recall, precision, F1-score and the confusion matrices are shown in Figure 6.6. The size of grid is determined based on the size of the dataset. For a dataset of size[7] n, the area of the grid is $2n$. The number of ants is considered 10% of dataset size, and the ant radius of perception r ranges from (1 to $\frac{1}{4}\sqrt{grid\ area}$. For the restaurant dataset, for example, the number of ants is 120 as the dataset size is 1200, grid dimension is 50×50.

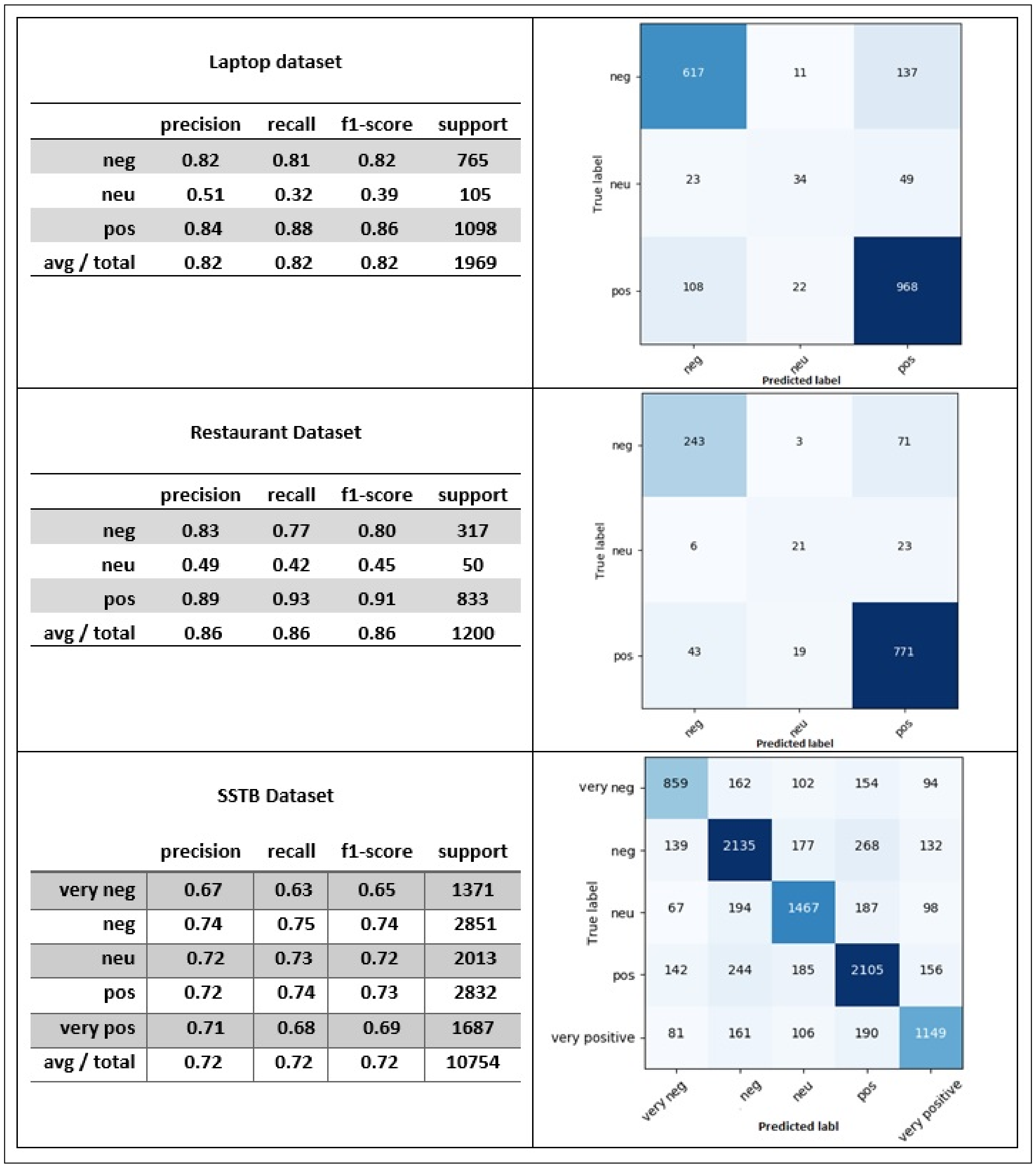
As I explained in section 5.4, the CACA-AR is evaluated using 10-fold cross-validation technique whereas only the training folds are used to generate pairwise constraints. To obtain the maximum accuracy, all possible CL and ML constraints are used during the clustering. For example, when the restaurant dataset is partitioned into 10 folds of size 120. The maximum number of constraints[8] that can generated from 9 training folds (1080 instances) is $291,330$. We can see that CACA-AR performs much better than logistic regression and lexicon-based approaches across all datasets as the average F1-scores indicate. For the laptop dataset, for instance, the CACA-AR scores are 0.81 and 0.88 recall values for positive and negative classes, respectively, with precision values of 0.82 and 0.84. This improvement is a result of increasing true positive values (617 and 968) and decreasing false negatives and false positives for both classes. However, this improvement, in terms of recall and precision is limited for the neutral class as the number of neutral sentences (105) is low with respect

[7]size of dataset: is the number of labeled instances given in the dataset

[8]In general, if the number of instances in training folds is n, the maximum number of pairwise constraints is $n(n-1)/4$

to the number of positive and negative sentences (765 and 1098). As a result, the number of ML and CL pairwise constraints, which relate neutral instances is much less than positive and negative instances. Therefore, we can see from the confusion matrix of laptop dataset that only 34 sentences out of 105 neutral sentences were correctly predicted. Similarly, CACA-AR outperforms baselines scores for restaurant dataset. However, it suffers from low recall and precision scores of the neutral class. This weakness is due to the low number of neutral sentences (50) with respect to the positive (813) and negative (317) instances. Consequently, the number of generated ML and CL constraints, which relate neutral instances is much less than the of number of constraints generated for positive and negative instances. Unlike logistic regression approach, which suffers from identifying very negative, neutral and very positive classes (Figure 6.7), CACA-AR scores significant results across all classes as the recall and precision values indicate. Notice that the number of instances in neutral class is much higher than laptop and restaurant dataset. Therefore, it achieves higher scores as more constraints can be generated for this class. This impact can also be seen in the very negative and very positive classes. There are two reasons for this improvement. First, constraints capture sentiment polarity, so the ants create clusters such that the sentences within each cluster tend to have the class label. Second, CACA-AR, as a semi-supervised approach, depends on calculating the cosine similarity between feature vectors for the sentences if the sentences is in the test data. Otherwise, the similarity is given by the constraints: either 1 (if both instances are linked by ML) or 0 (both instances are linked by CL)

As a clustering technique, CACA-AR can find clusters within each class of data and determine how the sentence patterns convey sentiment information. It does not require training such as the fully supervised classifiers.



Laptop dataset

	precision	recall	f1-score	support
neg	0.82	0.81	0.82	765
neu	0.51	0.32	0.39	105
pos	0.84	0.88	0.86	1098
avg / total	0.82	0.82	0.82	1969

True label \ Predicted label	neg	neu	pos
neg	617	11	137
neu	23	34	49
pos	108	22	968

Restaurant Dataset

	precision	recall	f1-score	support
neg	0.83	0.77	0.80	317
neu	0.49	0.42	0.45	50
pos	0.89	0.93	0.91	833
avg / total	0.86	0.86	0.86	1200

True label \ Predicted label	neg	neu	pos
neg	243	3	71
neu	6	21	23
pos	43	19	771

SSTB Dataset

	precision	recall	f1-score	support
very neg	0.67	0.63	0.65	1371
neg	0.74	0.75	0.74	2851
neu	0.72	0.73	0.72	2013
pos	0.72	0.74	0.73	2832
very pos	0.71	0.68	0.69	1687
avg / total	0.72	0.72	0.72	10754

True label \ Predicted labl	very neg	neg	neu	pos	very positive
very neg	859	162	102	154	94
neg	139	2135	177	268	132
neu	67	194	1467	187	98
pos	142	244	185	2105	156
very positive	81	161	106	190	1149

Figure 6.6: F1-score, Precision and Recall - CACA-AR Approach: very neg: very negative class, neg: negative class, neu: neutral class, pos: positive, very pos: very positive

6.8 Results of Aspect Category Identification

This section presents the results of aspect category identification for laptop and restaurant datasets. For both datasets, the goal is to assign each sentence with one of the aspect category labels. The aspect category labels for each dataset are listed in Tables (6.8;6.9) along with the number of sentences within each category (support). For example, for laptop dataset, there are 304 sentences labeled with the aspect category HARDWARE. Unlike sentiment prediction task, the task of aspect category identification does not depend on using lexicons. Therefore, the only baseline for this task is the logistic regression approach. In section 6.8.1, I present the results of logistic regression approach for this task while the results of CACA-AR are presented in section 6.8.2.

Table 6.8: Aspect Categories for Laptop Dataset

LAPTOP	HARDWARE	SOFTWARE	COMPANY	GRAPHICS	total
1224	304	118	199	124	1969

Table 6.9: Aspect Categories for Restaurant Dataset

RESTAURANT	SERVICE	FOOD	DRINKS	AMBIENCE	LOCATION	total
324	196	496	40	126	18	1200

The distribution of sentences in both datasets is highly unbalanced. In the laptop dataset, for example, the dominant class is *laptop* with 1224 sentences. Generally, the aspect category in such sentences is about entity itself, not any of its aspects or attributes. Similarly, the dominant class in the restaurant dataset is *restaurant* with

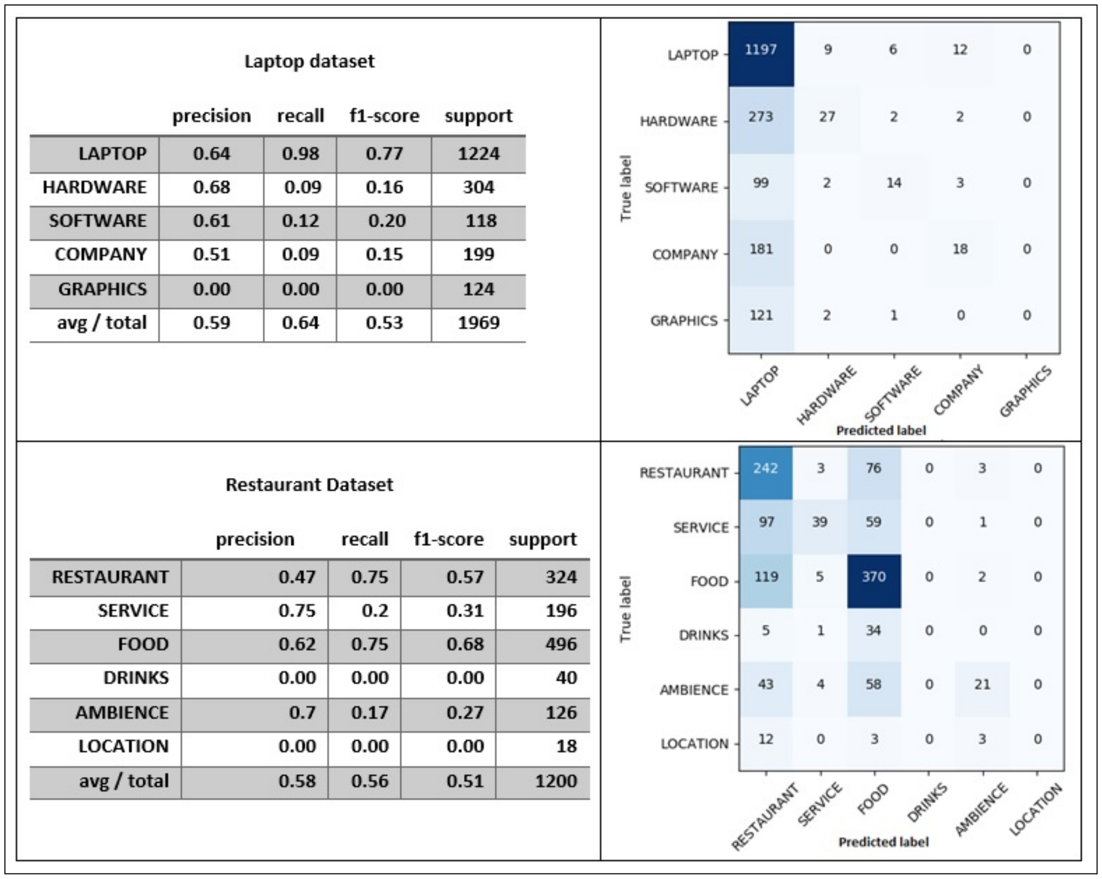
324 sentences, which describe a restaurant in general.

6.8.1 Logistic Regression Approach

The recall, precision and F1-score scores of logistic regression are shown in Figure 6.7. The logistic regression classifier is trained using paragraph vector model, as described in section 6.3 . The scores for each dataset is calculated by 10-fold cross validation technique. For the laptop dataset, the average F1-score is 0.53.

However, a major weakness in this approach is that the classifier tend to incorrectly predict most frequent aspect category “laptop” as a label for the rest of the classes. For example, out of the 304 sentences about “hardware”, 273 sentences were incorrectly predicted as *laptop*. This same weakness can be seen in “software”, “company” and “graphics”. This is due to the fact using paragraph vector model to generate feature vectors results in highly overlapped classes, especially when aspect categories are semantically-related. For example, the aspect categories *laptop*, “hardware” and “software” are semantically-related. As a result of this, the recall score of *laptop* class is high (0.87), while the precision is low.

For the restaurant dataset, as shown in Figure 6.7, the LR classifier tends to incorrectly predict sentences of infrequent classes by labeling them as either “restaurant” or “food” (labels of frequent classes). Therefore the recall scores of “restaurant” and “food” are the highest, but their precision scores are low.



Laptop dataset

	precision	recall	f1-score	support
LAPTOP	0.64	0.98	0.77	1224
HARDWARE	0.68	0.09	0.16	304
SOFTWARE	0.61	0.12	0.20	118
COMPANY	0.51	0.09	0.15	199
GRAPHICS	0.00	0.00	0.00	124
avg / total	0.59	0.64	0.53	1969

	LAPTOP	HARDWARE	SOFTWARE	COMPANY	GRAPHICS
LAPTOP	1197	9	6	12	0
HARDWARE	273	27	2	2	0
SOFTWARE	99	2	14	3	0
COMPANY	181	0	0	18	0
GRAPHICS	121	2	1	0	0

Restaurant Dataset

	precision	recall	f1-score	support
RESTAURANT	0.47	0.75	0.57	324
SERVICE	0.75	0.2	0.31	196
FOOD	0.62	0.75	0.68	496
DRINKS	0.00	0.00	0.00	40
AMBIENCE	0.7	0.17	0.27	126
LOCATION	0.00	0.00	0.00	18
avg / total	0.58	0.56	0.51	1200

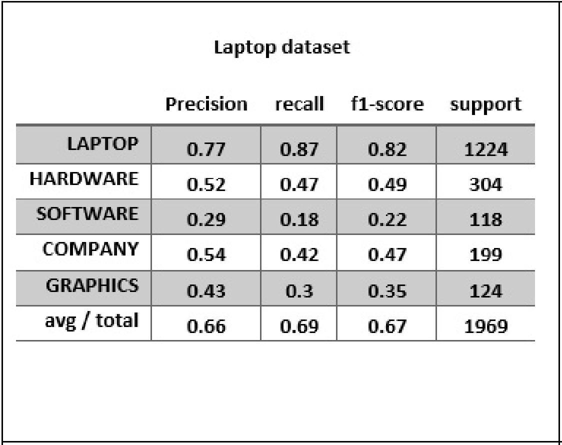
	RESTAURANT	SERVICE	FOOD	DRINKS	AMBIENCE	LOCATION
RESTAURANT	242	3	76	0	3	0
SERVICE	97	39	59	0	1	0
FOOD	119	5	370	0	2	0
DRINKS	5	1	34	0	0	0
AMBIENCE	43	4	58	0	21	0
LOCATION	12	0	3	0	3	0

Figure 6.7: F1-score, Precision and Recall - Logistic Regression Approach

6.8.2 CACA-AR Approach

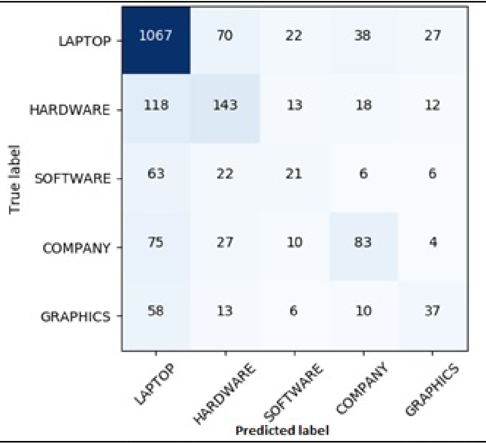
The results of CACA-AR are shown in Figure 6.8. The scores are 10-fold cross-validated. All possible ML and CL constraints are generated from training folds while the scores are calculated for the test fold. Similar to the logistic regression, CACA-AR tends to incorrectly assign the most frequent class label "laptop" to the rest of the classes as shown in the laptop confusion matrix. This results in increasing recall value for the laptop class but decreasing precision at the same time. However, we can see that the precision score of laptop outperforms logistic regression due to incorporating ML and CL constraints. This provides a better identification for the infrequent classes such as “software”, “company” and “graphics” as the average precision score for ACA-AR is higher than logistic regression. In addition, we can see that CACA-AR obtains 0.35 F1-score for the “graphics” class which is entirely undetected by logistic regression.

Similar to the results of laptop dataset, we can see that CACA-AR tends to incorrectly assign frequent class labels such as “restaurant” and “food” to the infrequent classes such as “drink” and “location”. However, incorporating pairwise constraint results in improving the precision for the restaurant dataset classes without affecting recall values. CACA-AR achieves 0.63 average precision while logistic regression achieve 0.58. This results in improving overall F1-score for this dataset.

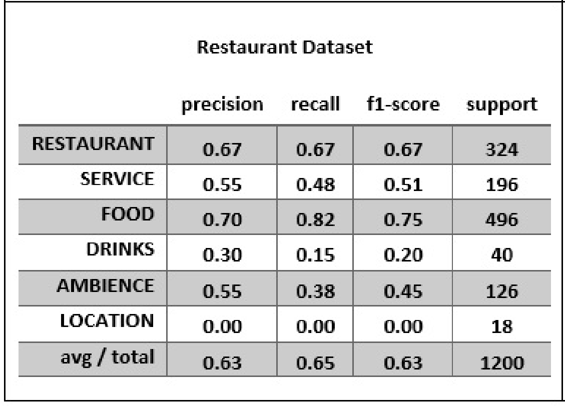


Laptop dataset

	Precision	recall	f1-score	support
LAPTOP	0.77	0.87	0.82	1224
HARDWARE	0.52	0.47	0.49	304
SOFTWARE	0.29	0.18	0.22	118
COMPANY	0.54	0.42	0.47	199
GRAPHICS	0.43	0.3	0.35	124
avg / total	0.66	0.69	0.67	1969

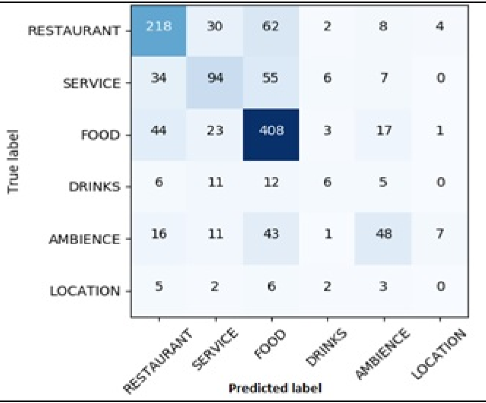


True label \ Predicted label	LAPTOP	HARDWARE	SOFTWARE	COMPANY	GRAPHICS
LAPTOP	1067	70	22	38	27
HARDWARE	118	143	13	18	12
SOFTWARE	63	22	21	6	6
COMPANY	75	27	10	83	4
GRAPHICS	58	13	6	10	37



Restaurant Dataset

	precision	recall	f1-score	support
RESTAURANT	0.67	0.67	0.67	324
SERVICE	0.55	0.48	0.51	196
FOOD	0.70	0.82	0.75	496
DRINKS	0.30	0.15	0.20	40
AMBIENCE	0.55	0.38	0.45	126
LOCATION	0.00	0.00	0.00	18
avg / total	0.63	0.65	0.63	1200



True label \ Predicted label	RESTAURANT	SERVICE	FOOD	DRINKS	AMBIENCE	LOCATION
RESTAURANT	218	30	62	2	8	4
SERVICE	34	94	55	6	7	0
FOOD	44	23	408	3	17	1
DRINKS	6	11	12	6	5	0
AMBIENCE	16	11	43	1	48	7
LOCATION	5	2	6	2	3	0

Figure 6.8: F1-score, Precision and Recall - CACA-AR Approach

In summary, although the state-of-the-art techniques for language models, such as paragraph vector model, can capture many grammatical and semantic features of word sequences, they are still insufficient for the tasks of ABSA. Since the learning of such feature representations is independent from class labels, they lack the required features to capture the sentiment or aspect category information. Therefore, using such feature representations in machine learning classification models is not sufficient for the tasks of ABSA. Learning feature representations for the tasks of ABSA is still a major challenge and an active area of research to improve the accuracy of ABSA.

Another challenge in sentiment analysis is the existence of neutral class (i.e., to determine whether a piece of text is subjective or objective (neutral)). This is because the instances of neutral class lies at the boundaries of negative and positive classes. To remedy these challenges, I adopted CACA-AR, as semi-supervised clustering approach. For the task of sentiment prediction, incorporating pairwise constraints (ML and CL) in ACA-AR enforced the sentences with the same class label to be in one cluster. Note that pairwise constraints are independent of feature representation because they are generated based on class labels.

A major advantage of CACA-AR is that, clustering is "guided" or "biased" by pairwise constraints. This allows us to use the same feature representations for both ABSA tasks. For example, if constraints are generated from sentiment labels, then the goal of clustering is to group text pieces with the same polarity within the same cluster. Similarly, when constraints are generated from aspect category labels, then the goal of clustering is to group text pieces with the same aspect category in the same cluster. This does not imply CACA-AR converges to a number of clusters that is equal to the number of classes in a given dataset. However, it finds a clustering solution (i.e., intrinsic structure) where most instances within one cluster have the same label. This can be achieved with CACA-AR because it does not require the number of clusters a priori. For example, clustering the sentences of positive-sentiment class can result in k clusters. The instances in each cluster share the same polarity as well as other grammatical and semantic features[9], which, in turn, can be analyzed to improve the quality of feature representation.

[9]e.g. sentiment analysis is highly dependent on the grammatical type of the sentence, such as conditional, imperative, declarative etc...

Moreover, CACA-AR clustering is significantly affected by distribution of the classes in a dataset. With larger class size, higher number of ML and CL constraints can be generated, and hence, better clustering quality can be achieved. This impact can be seen in both laptop and restaurant datasets for both ABSA tasks. CACA-AR is highly sensitive to the values of hyper-parameters, such as the number of ants to be used, the size of grid and the optimal value of radius of perception. Fine-tuning these parameters is problematic as the algorithm is computationally-intensive. In addition, although there is no guarantee that CACA-AR can find an optimal clustering solution, it can achieve a Pareto-optimal solution [10] with the adaptive radius strategy.

[10]is a clustering solution where intra-cluster similarity and inter-cluster dissimilarity are simultaneously maximized

Chapter 7

Conclusion

7.1 Conclusion

In this thesis, I presented Ant Brood Clustering Algorithm with Adaptive Radius (ACA-AR), a new variant of ACA that involves three major enhancements over existing ACA model. (i) ACA-AR employs Kernel density estimation and Sigmoid function to measure average dissimilarity of data objects within ants neighborhood and to estimate ants pick-up and drop-off probabilities; (ii) it uses memoryless, tireless ants; (iii) it allows each ant to adapt its radius of perception so that ants collectively can avoid the convergence to a local-optimum solution. I have experimentally validated ACA-AR on three benchmark data sets that present different clustering challenges. The results revealed that ACA-AR outperforms most frequently used clustering algorithms, such as ACA, Mean Shift and k-means in terms of clustering accuracy, completeness, and homogeneity. The results of parallel implementation of ACA-AR have shown that a speedup up to 39x can be obtained compared to the sequential counterpart using GPU.

Following the same intuition of ACA, I extended ACA-AR into CACA-AR. CACA-AR is a modified semi-supervised variant of ACA-AR for clustering multidimensional data in the presence of instance-level pairwise constraints (must-linked and cannot-linked). Unlike ACA-AR model, CACA-AR takes the advantage of pairwise constraints to further improve the estimation of ant pickup and drop-off probabilities. I experimentally validated CACA-AR with different sets pairwise constraints on three benchmark data sets that present different clustering challenges. The results have also shown that the accuracy, completeness, and homogeneity of ACA-AR substantially improved when pairwise constraints were incorporated, especially to high-dimensional datasets.

Finally, I evaluated the application of CACA-AR to the tasks of Aspect-based Sentiment Analysis in the domain of product reviews, aspect identification and sentiment prediction. I compared CACA-AR as a semi-supervised with two common baseline approaches on three benchmark datasets: (i) Multi-class Logistic Regression as fully-supervised approach. (ii) Lexicon-based approach as an unsupervised approach. The results revealed that CACA-AR outperforms both baselines by a large-margin (20% in terms of F1-score), which demonstrates CACA-AR effectiveness to real-world datasets for ABSA.

In conclusion, ACA-AR as unsupervised clustering algorithm has many advantages over traditional clustering algorithms: (i) it allows the ants to explore (search) a significantly larger number of candidate clustering solutions; (ii) it is more capable of detecting outliers and finding the exact number of clusters in the data; (iii) it substantially improves the spatial separation of clusters on the grid which is an essential requirement to retrieve the clusters. Similarly, CACA-AR as a semi-supervised clustering algorithm can significantly improve clustering accuracy in three circumstances: (i) clustering high-dimensional data whereas "curse of dimensionality" is a major obstacle to improve accuracy; (ii) classifying data in domains whereby labeled data is limited and/or costly to acquire; (iii) clustering/classifying data in domains whereby data representation (feature vectors) does not capture data similarities efficiently.

7.2 Future Work

This thesis opens an interesting research avenues to improve ACA-AR and to apply CACA-AR to real-world applications:

- Since ant behavior is collective, a slight improvement in local density estimation within ant neighborhood could results in a large improvement in the outcome. Therefore, choosing appropriate density estimation technique is crucial to further improve ACA performance. For instance, multivariate density can be estimated via copulas.
- In my experiments, I fixed the ant radius of perception to a threshold where the ant covers 1/4 of grid area. Further analysis is required to determine the exact value of radius that guarantee achieving maximum solution quality with minimum computational time.
- In ACA-AR, there exist two parameters to improve solution quality, ant radius and kernel density estimate. the analysis should be extended to show the impact

of each in solution quality and computational time.

- In nature, ant brood sorting behavior is triggered or “guided” by pheromones, chemical substances that ants release for many purposes (e.g. signal danger to the colony, give directions about food location and attract mates). According to ecological findings in ant-brood behavior, ants mark objects (e.g. grains, broods and larvae) with pheromones when they are picked up. Ants tend to pick up objects that are previously marked with pheromones, probably due to the high concentration of pheromone. Ants pick up and drop off objects at constant rates. To my knowledge, these features are not captured in existing ACA models.

- In the existing ACA models, ants move randomly in 2D-grid. This limits ant movement to eight directions at maximum. Therefore, it is interesting to consider grids in higher dimensional space.

In terms of real-world applications of CACA-AR:

- The algorithm is highly applicable to the community detection problem in complex networks. There are two reasons for this: (i) CACA-AR does not make any assumption about the number of communities (clusters) in the network; (ii) CACA-AR can take advantage of prior knowledge (must-linked and cannot-linked constraints) to guide the detection process.

- ACA-AR is applicable to multi-objective optimization problems. For example, the portfolio optimization in Finance requires optimizing two contradicting objectives: choosing portfolio assets that maximize expected value returns, but minimize risk, often measured by the standard divination of portfolio returns at same time.

In general, ant algorithms are suitable for many dynamic applications.

Bibliography

C. C. Aggarwal and C. K. Reddy. *Data clustering: algorithms and applications*. Chapman and Hall/CRC, 2013.

S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204, 2010.

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(Jun):937–965, 2005.

J. Barnes, R. Klinger, and S. S. i. Walde. Assessing state-of-the-art sentiment models on state-of-the-art sentiment datasets. *arXiv preprint arXiv:1709.04219*, 2017.

S. Basu, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. In *SDM*, volume 4, pages 333–344. SIAM, 2004a.

S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68. ACM, 2004b.

S. Basu, I. Davidson, and K. Wagstaff. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press, 2008.

M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48. ACM, 2003.

M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 11. ACM, 2004.

S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. A. Reis, and J. Reynar. Building a sentiment summarizer for local service reviews. In *WWW Workshop on NLP in the Information Explosion Era*, volume 14, pages 339–348, 2008.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308, 2003.

E. Boiy and M.-F. Moens. A machine learning approach to sentiment analysis in multilingual web texts. *Information retrieval*, 12(5):526–558, 2009.

Y. Choi and C. Cardie. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 590–598. Association for Computational Linguistics, 2009.

D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, 4(1):17–32, 2003.

I. Davidson and S. Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 59–70. Springer, 2005a.

I. Davidson and S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 138–149. SIAM, 2005b.

A. De Mauro, M. Greco, and M. Grimaldi. What is big data? a consensual definition and a review of key research topics. In *AIP conference proceedings*, volume 1644, pages 97–104. AIP, 2015.

J.-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien. The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 356–363, 1991.

T. Duong and M. L. Hazelton. Cross-validation bandwidth matrices for multivariate kernel density estimation. *Scandinavian Journal of Statistics*, 32(3):485–506, 2005.

D. Eisenberg, E. M. Marcotte, I. Xenarios, and T. O. Yeates. Protein function in the post-genomic era. *Nature*, 405(6788):823–826, 2000.

A. Esuli and F. Sebastiani. Sentiwordnet: a high-coverage lexical resource for opinion mining. *Evaluation*, pages 1–26, 2007.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975.

F. W. Glover and G. A. Kochenberger. *Handbook of metaheuristics*, volume 57. Springer Science & Business Media, 2006.

Z. Hai, K. Chang, and J.-j. Kim. Implicit feature identification via co-occurrence association rule mining. In *Computational Linguistics and Intelligent Text Processing*, pages 393–404. Springer, 2011.

J. Handl and B. Meyer. Improved ant-based clustering and sorting in a document retrieval interface. In *International Conference on Parallel Problem Solving from Nature*, pages 913–923. Springer, 2002.

Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

J. B. Hirschberg and A. Rosenberg. V-measure: A conditional entropy-based external cluster evaluation. Proceedings of EMNLP, 2007.

T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

C. Honghao, F. Zuren, and R. Zhigang. Community detection using ant colony optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 3072–3078. IEEE, 2013.

P. Horton, K.-J. Park, T. Obayashi, N. Fujita, H. Harada, C. Adams-Collier, and K. Nakai. Wolf psort: protein localization predictor. *Nucleic acids research*, 35 (suppl 2):W585–W587, 2007.

M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

N. Jakob and I. Gurevych. Extracting opinion targets in a single-and cross-domain setting with conditional random fields. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1035–1045. Association for Computational Linguistics, 2010.

L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics, 2011.

W. Jin, H. H. Ho, and R. K. Srihari. A novel lexicalized hmm-based learning framework for web opinion mining. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 465–472. Citeseer, 2009.

K. Kamvar, S. Sepandar, K. Klein, D. Dan, M. Manning, and C. Christopher. Spectral learning. In *International Joint Conference of Artificial Intelligence*. Stanford InfoLab, 2003.

J. Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. 2002.

N. Kobayashi, K. Inui, and Y. Matsumoto. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *EMNLP-CoNLL*, volume 7, pages 1065–1074. Citeseer, 2007.

H. W. Kuhn. Nonlinear programming: a historical view. *ACM SIGMAP Bulletin*, (31):6–18, 1982.

H. Lakkaraju, C. Bhattacharyya, I. Bhattacharya, and S. Merugu. Exploiting coherence for the simultaneous discovery of latent facets and associated sentiments. In *SDM*, pages 498–509. SIAM, 2011.

J. H. Lau and T. Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.

F. Li, C. Han, M. Huang, X. Zhu, Y.-J. Xia, S. Zhang, and H. Yu. Structure-aware review mining and summarization. In *Proceedings of the 23rd international conference on computational linguistics*, pages 653–661. Association for Computational Linguistics, 2010.

Z. Li, M. Zhang, S. Ma, B. Zhou, and Y. Sun. Automatic extraction for product feature words from comments on the web. In *Information Retrieval Technology*, pages 112–123. Springer, 2009.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

B. Liu. *Research Projects*, 2009 (accessed May 5, 2016). URL https://www.cs.uic.edu/~liub/.

B. Liu. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press, 2015.

B. Liu, M. Hu, and J. Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351. ACM, 2005.

Y. Y. Liu and Y. Y. Liu. A polymorphic ant-based algorithm for graph clustering. 2016.

C. Long, J. Zhang, and X. Zhut. A review selection approach for accurate feature rating estimation. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 766–774. Association for Computational Linguistics, 2010.

B. Lu, M. Ott, C. Cardie, and B. K. Tsou. Multi-aspect sentiment analysis with topic models. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 81–88. IEEE, 2011.

E. D. Lumer and B. Faieta. Diversity and adaptation in populations of clustering ants. In *Proceedings of the third international conference on Simulation of adaptive behavior: from animals to animats 3: from animals to animats 3*, pages 501–508. MIT Press, 1994.

K. Lund and C. Burgess. Hyperspace analogue to language (hal): A general model semantic representation. In *Brain and Cognition*, volume 30, pages 5–5. ACADEMIC PRESS INC JNL-COMP SUBSCRIPTIONS 525 B ST, STE 1900, SAN DIEGO, CA 92101-4495, 1996.

J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

Q. Mei, X. Ling, M. Wondra, H. Su, and C. Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, pages 171–180. ACM, 2007.

P. D. Michailidis and K. G. Margaritis. Accelerating kernel density estimation on the gpu using the cuda framework. *Applied Mathematical Sciences*, 7(30):1447–1476, 2013.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

M. T. Moore. Twitter index tracks sentiment on obama, romney, 2012. URL http://usatoday30.usatoday.com/news/politics/story/2012-08-01/twitter-political-index/56649678/1. Accessed Oct. 18, 2016.

S. Navlakha and Z. Bar-Joseph. Distributed information processing in biological and computational systems. *Communications of the ACM*, 58(1):94–102, 2015.

K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000.

B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 27–35, 2014.

M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado*, pages 486–495, 2015.

M. Pourrajabi, D. Moulavi, R. J. G. B. Campello, A. Zimek, J. Sander, R. Goebel, et al. Model selection for semi-supervised clustering. In *International Conference on Extending Database Technology, 17.* Athens, 2014.

M. Qasem, Y. Y. Liu, Z. Wang, P. Thulasiraman, and R. K. Thulasiram. Enhancing ant brood clustering with adaptive radius of perception and non-parametric estimation on multi-core architectures. In *International Conference on Intelligent Networking and Collaborative Systems*, pages 301–312. Springer, 2017.

G. Qiu, B. Liu, J. Bu, and C. Chen. Opinion word expansion and target extraction through double propagation. *Computational linguistics*, 37(1):9–27, 2011.

S. Raju, P. Pingali, and V. Varma. An unsupervised approach to product attribute extraction. In *Advances in Information Retrieval*, pages 796–800. Springer, 2009.

R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP*

Frameworks, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

D. L. Rohde, L. M. Gonnerman, and D. C. Plaut. An improved model of semantic similarity based on lexical co-occurrence. *Communications of the ACM*, 8:627–633, 2006.

P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

J. Ruppenhofer, S. Somasundaran, and J. Wiebe. Finding the sources and targets of subjective expressions. In *LREC*, 2008.

C. Scaffidi, K. Bierhoff, E. Chang, M. Felker, H. Ng, and C. Jin. Red opal: product-feature scoring from reviews. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 182–191. ACM, 2007.

scikit-learn developers. Clustering. http://scikit-learn.org/stable/modules/clustering.html#mean-shift. Accessed: 2017-02-05.

M. Senoussaoui, P. Kenny, P. Dumouchel, and T. Stafylakis. Efficient iterative mean shift based cosine dissimilarity for multi-recording speaker clustering. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7712–7715. IEEE, 2013.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

R. Socher, C. D. Manning, and A. Y. Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9, 2010.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

P. J. Stone, D. C. Dunphy, and M. S. Smith. The general inquirer: A computer approach to content analysis. 1966.

I. Titov and R. McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120. ACM, 2008.

P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

M. S. Vohra and J. Teraiya. Applications and challenges for sentiment analysis: A survey. In *International Journal of Engineering Research and Technology*, volume 2. ESRSA Publications, 2013.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.

B. Liu. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2:627–666, 2010.

K. Wagstaff and C. Cardie. Clustering with instance-level constraints. *AAAI/IAAI*, 1097, 2000.

K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

J. Wang, E. Osagie, P. Thulasiraman, and R. K. Thulasiram. Hopnet: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. *Ad Hoc Networks*, 7(4):690–705, 2009.

X. Wang and I. Davidson. Flexible constrained spectral clustering. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 563–572. ACM, 2010.

X. Wang, B. Qian, and I. Davidson. On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery*, pages 1–30, 2014.

Y.-C. Wei and C.-K. Cheng. Ratio cut partitioning for hierarchical designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(7): 911–921, 1991.

W. J. Welch. Algorithmic complexity: three np-hard problems in computational statistics. *Journal of Statistical Computation and Simulation*, 15(1):17–25, 1982.

J. Wiebe, T. Wilson, and C. Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210, 2005.

B. Wu and Z.-Z. Shi. An ant colony algorithm based partition algorithm for tsp. *CHI-*

NESE JOURNAL OF COMPUTERS-CHINESE EDITION-, 24(12):1328–1333, 2001.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15:505–512, 2003.

X.-H. Xu, Z.-J. Pan, P. He, and L. Chen. Constrained ant clustering. In *ICMLC*, pages 1566–1570, 2011.

Y. Yang, H. Wang, C. Lin, and J. Zhang. Semi-supervised clustering ensemble based on multi-ant colonies algorithm. In *International Conference on Rough Sets and Knowledge Technology*, pages 302–309. Springer, 2012.

Z. Zhai, B. Liu, H. Xu, and P. Jia. Clustering product features for opinion mining. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 347–354. ACM, 2011.

Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, 2004.

X. Zhu. Semi-supervised learning literature survey. 2005.