DOCTORAL DISSERTATION

COLLABORATIVE RANKING AND COLLABORATIVE CLUSTERING

by

Zheng Chen

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2013

This manuscript has been read and accepted for the
Graduate Faculty in Computer Science in satisfaction of the
dissertation requirement for the degree of Doctor of Philosophy.

Heng Ji

Date

Chair of Examining Committee

Ted Brown

Date

Executive Officer

Andrew Rosenberg

Robert Haralick

Liang Huang

Radu Florian

Wen Wang

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

COLLABORATIVE RANKING AND COLLABORATIVE CLUSTERING

by

Zheng Chen

Adviser: Professor Heng Ji

Ranking and clustering are two important problems in machine learning and have wide applications in Natural Language Processing (NLP). A ranking problem is typically formulated as ranking a collection of candidate "objects" with respect to a "query" while a clustering problem is formulated as organizing a set of instances into groups such that members in each group share some similarity while members across groups are dissimilar. In this thesis, we introduce collaborative schemes into ranking and clustering problems, and name them as "collaborative ranking" and "collaborative clustering" respectively. Contrast to the tradition non-collaborative schemes, collaborative ranking leverages strengths from multiple query collaborators and ranker collaborators while collaborative clustering leverages strengths from multiple instance collaborators and clusterer collaborators. We select several typical NLP problems as our case studies including entity linking, document clustering and name entity clustering.

Dedicated to the memories of my father Canyan Chen.

Acknowledgements

First of all, I am grateful to my advisor, Prof. Heng Ji, for her tremendous guidance, enthusiasm, dedication when she started advising me in the Fall of 2008. Without her, I would never have been so addicted to the art of scientific research in the area of computational linguistics. I am indebted to her for all her patient guidance through the writing of this thesis. She also awarded me a honorable title as "her first Ph.D. student". Because of her, I have the greatest opportunities to participate in large-scale project evaluations such as Knowledge Base Population (KBP). The tasks in KBP have greatly motivated my thesis work, and they served as two case studies in my thesis.

I want to thank the other five committee members Prof. Robert Haralick, Prof. Andrew Rosenberg, Prof. Liang Huang, Dr. Radu Florian, Dr. Wen Wang who have put great efforts in supervising my thesis. Special thanks to Prof. Robert Haralick who has also supervised my course work of Pattern Recognition in the early stage of my Ph.D. study. His insights in machine learning have also greatly inspired my research in my thesis.

I would like to thank Dr. Radu Florian again for his supervision when I did a wonderful summer internship at IBM in 2011. Also thanks to the other two researchers at IBM, Dr. Hema Raghavan and Dr. Vittorio Castelli who provided great guidance during the internship.

I thank all the other current or former lab members from BLENDER lab, they are Haibo Li, Arkaitz Zubiaga, Hao Li, Qi Li, Taylor Cassidy, Hongzhao Huang, Adam Lee, Suzanne Tamang, Javier Artiles, Matt Snover, Wen-Pin (Daniel) Lin, Xiang Li, Sam Anzaroot, Mingmin Ge, Kurt Xu, Juan Liu, and Prashant Gupta. I will never forget the memorable days in the lab, sharing the happiness and enjoyable

moments. Special thanks to Haibo Li, Qi Li and Hongzhao Huang who I have worked shoulder to shoulder with in the BOLT evaluation, Adam Lee, Suzanne Tamang and Xiang Li who I have worked shoulder to shoulder with in the KBP evaluation. Without their great efforts, we could not have completed those great projects.

I am most grateful for my beloved girlfriend Sumeng Guo and my future mother-in-law Mengxia Xing, for their caring, patience, support and love. I am also most indebted to my beloved mother Baozhen Chen and my beloved passed father Canyan Chen. They take pride in me, and I will be always proud to be their son, forever.

This work was supported by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053, the U.S. NSF Grants IIS-0953149 and IIS-1144111 and the U.S. DARPA BOLT program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Contents

Contents v

List of Tables x

List of Figures xii

1 Introduction 1

1.1 Collaborative Ranking 2

1.2 Collaborative Clustering 6

1.3 Terminology 9

1.4 Thesis Structure 11

1.5 Related Publications 12

2 Preliminaries 13

2.1 Ranking 13

2.1.1 Problem Formulation 13

2.1.2 Non-collaborative Ranking Approaches 14

2.1.3 Query Expansion 16

2.1.4 Ensemble Ranking 16

2.2 Clustering 17

2.2.1 Problem Formulation 17

2.2.2 Non-collaborative Clustering Approach 18

3 Collaborative Ranking 20

3.1 Micro Collaborative Ranking(MiCR) 20
3.2 Macro Collaborative Ranking(MaCR) 22
3.3 Micro-Macro Collaborative Ranking (MiMaCR) 23
3.4 Macro-Micro Collaborative Ranking (MaMiCR) 24
3.5 Summary 25

4 Entity Linking: A Case Study of Collaborative Ranking 26
4.1 Problem Formulation 26
4.2 Related Research 26
4.3 General Framework 27
4.4 Baseline Rankers 28
4.5 MiCR for Entity Linking 29
4.6 MaCR for Entity Linking 29
4.7 MiMaCR for Entity Linking 30
4.8 Experimental Results 30
4.8.1 Data and Evaluation Metric 30
4.8.2 Performance of 8 Baseline Rankers 31
4.8.3 Impact of MiCR 32
4.8.4 Impact of MaCR 33
4.8.5 Impact of MiMaCR 34
4.9 Summary 36

5 Collaborative Clustering 37
5.1 Motivations 38
5.1.1 Why Collaborative Instances Can Help? 38
5.1.2 Why Collaborative Clusterers Can Help? 39
5.2 Clustering Validation 40
5.2.1 Two Basic Criteria for a Good Internal Measure 41
5.2.2 Overview of Internal Validation Measures 42
5.2.3 Comparison of Various Internal Measures 50

5.3 Micro Collaborative Clustering (MiCC) . . . 52
5.3.1 Mechanism of Populating Collaborative Instances . . . 52
5.3.2 Selection of Collaborative Instances . . . 52
5.3.3 Algorithms of MiCC . . . 53
5.3.4 Obtaining Final Clustering Solution . . . 53
5.4 Macro Collaborative Clustering (MaCC) . . . 54
5.4.1 Ensemble Generation . . . 54
5.4.1.1 K-means algorithm . . . 55
5.4.1.2 Agglomerative clustering algorithms[84] . . . 56
5.4.1.3 Partitional clustering algorithms[84] . . . 57
5.4.2 Consensus Functions . . . 58
5.4.2.1 Consensus function based on mutual information . . . 59
5.4.2.2 Consensus function based on co-association matrix . . . 60
5.4.2.3 Consensus function based on voting . . . 61
5.4.2.4 Consensus function based on graph formulation . . . 62
5.5 Micro Macro Collaborative Clustering (MiMaCC) . . . 64
5.6 Summary . . . 64

6 Document Clustering: A Case Study of Collaborative Clustering 65

6.1 Introduction . . . 65
6.2 Problem Formulation . . . 66
6.3 Distance Functions . . . 67
6.4 External Clustering Validation Measures . . . 69
6.5 Experiments . . . 75
6.5.1 Data Sets . . . 75
6.5.2 Experimental Tools . . . 76
6.5.3 Impact of External Measures . . . 76
6.5.4 Impact of Similarity Functions . . . 77
6.5.5 Impact of Baseline Clustering algorithms . . . 78
6.5.6 Correlation between Internal Measures and V-Measure . . . 84

6.5.7 Impact of MiCC . . . 86
6.5.8 Impact of MaCC . . . 88
6.5.9 Impact of MiMaCC . . . 94
6.6 Summary . . . 94

7 Name Entity Clustering: Another Case Study of Collaborative Clustering 101
7.1 Problem Formulation . . . 101
7.2 Related Research . . . 102
7.3 Name Variation Identification . . . 103
7.3.1 Problem Formulation . . . 103
7.3.2 Approach . . . 103
7.3.3 Dataset and Measure . . . 106
7.3.4 Experimental Results . . . 107
7.4 Name Disambiguation Detection . . . 110
7.4.1 Problem Formulation . . . 110
7.4.2 Preprocessing Step . . . 110
7.4.3 Features . . . 113
7.4.4 Classification Model . . . 113
7.4.5 Dataset and Measure . . . 114
7.4.6 Experimental Results . . . 116
7.4.6.1 Feature Contribution . . . 116
7.4.6.2 Separate Models Vs. Single Model . . . 118
7.4.6.3 Impact of Separate Models with Reduced Features . . . 118
7.4.6.4 Impact of Handling Acronyms . . . 119
7.5 Name Entity Clustering . . . 119
7.5.1 Two Baseline systems . . . 119
7.5.2 Clustering Algorithms and Similarity Functions . . . 119
7.5.3 Experiments . . . 120
7.5.4 Dataset and Evaluation Metric . . . 120
7.5.4.1 Impact of Clustering Algorithms . . . 123

7.5.4.2 Impact of MiCC 133
7.5.4.3 Impact of MaCC 134
7.6 Summary 139

8 Conclusions and Future Directions **140**
8.1 Conclusions 140
8.2 Future Directions 143
8.2.1 Coupling Collaborative Ranking and Collaborative Clustering 143
8.2.2 Enhancing MiCR 144
8.2.3 Enhancing MaCR 145
8.2.4 Enhancing MiCC 148
8.2.5 A Third Application of Collaborative Clustering: Event Coreference Resolution 148

References **152**

List of Tables

2.1 Comparison of pointwise, pairwise and listwise ranking approaches. 15

4.1 Training, development and testing corpus. 30

4.2 Comparison of 8 baseline rankers. 31

5.1 Notations of internal validation measures 42

5.2 Summary of 12 internal validation measures 43

6.1 8 sample data sets from *hitech* data set 77

6.2 Correlation between DCV and the measures 77

6.3 Impact of four distance functions (clustering evaluated by V-measure) 78

6.4 Performance of 21 baseline clustering algorithms evaluated by V-measure . . . 78

6.5 Significance matrix among 21 clustering algorithms on dataset *k1b* 78

6.6 Significance matrix among 21 clustering algorithms on dataset *reviews* 79

6.7 Significance matrix among 21 clustering algorithms on dataset *sports* 79

6.8 Significance matrix among 21 clustering algorithms on dataset *hitech* 79

6.9 Comparisons of observations among [100, 101] and ours. 82

6.10 Performance of 12 clustering methods (k=10, 10-way clustering) on three datasets evaluated by F-score, reported in [99]. 82

6.11 Performance of 12 clustering methods (k=10) on three datasets evaluated by entropy, reported in [99]. 82

6.12 DCV of 21 baseline clustering algorithms 82

6.13 Cluster distribution generated by various clustering methods. cid: cluster id; size: number of documents in this cluster; heal,spor,busi,poli,ente,tech are 6 class names 84
6.14 Impact of 12 internal measures in determining the number of clusters k 85
6.15 Correlation between internal measures and V-measure 86
6.16 Average number of iterations that is needed to optimize Silhouette Coefficient . 88
6.17 Summary of MaCC experiments 92

7.1 Confusion matrix of name variant classifier. 107
7.2 Impact of various resources and rules in the task of name variation detection . . 107
7.3 An example of preprocessing a document associated with an instance 112
7.4 Features of maximum entropy model for name disambiguation 114
7.5 Class distribution of dataset1 and dataset2. 115
7.6 Confusion matrix of name disambiguation classifier. 115
7.7 Feature contribution ranked by information gain from high to low. 117
7.8 Performance of supervised classifier. 118
7.9 Examples of performance of one-in-one system 123
7.10 Comparison of 21 clustering algorithms 127
7.11 DCV values of 21 clustering algorithms for all names (known K) 129
7.12 DCV values of 21 clustering algorithms for all names (unknown K) 129
7.13 Comparison of similarity functions 132
7.14 Comparison of system generated K and prior K 133

List of Figures

1.1 Non-collaborative ranking and four collaborative ranking approaches. 4

1.2 An example of entity linking task. 6

1.3 Two motivating examples for instance-level collaborative clustering 7

1.4 An example of clusterer-level collaborative clustering. 8

1.5 An example of name entity clustering. 10

4.1 Framework of entity linking system. 27

4.2 MiCR: comparison of average, max, and min functions with Graph and Agglomerative (Aggr)-based query collaborator searching strategies (*tfidf* ranker). 32

4.3 MaCR: comparison of voting and average. 34

4.4 MaCR: voting function applied to the top 10 KBP2009 entity linking systems. . 35

4.5 MiMaCR: Comparison of MiMaCR and three supervised versions of g_1 (ListNet, Maxent, and SVM respectively). 35

5.1 Examples showing good and bad collaborative instances 37

5.2 An example of consensus function based on co-association matrix 61

5.3 An example of consensus function based on voting 63

6.1 Summary of external clustering measures by types 69

6.2 Illustrations of four formal constraints [1] 70

6.3 Satisfaction of constraints for various external measures [1], measures noted by * are extensions presented in [13] 71

6.4 Relations among entropy, conditional entropy, joint entropy and mutual information 73

6.5 Summary of datasets used for our experiments of document clustering 76
6.6 Impact of MiCC algorithm on the dataset *k1b* 87
6.7 Impact of MiCC algorithm on the dataset *reviews* 87
6.8 Impact of MiCC algorithm on the dataset *sports* 88
6.9 Impact of MaCC by incrementally adding clusterers using 3 serial similarity functions: cosine similarity (cos), correlation similarity (cor) and Jaccard similarity (jac) 96
6.10 Impact of MaCC by incrementally adding clusterers using 4 serial algorithms: repeated bisectional (rbr), direct k-way (direct), agglomerative (agglo) and K-means (kmeans) 97
6.11 Impact of MaCC by incrementally adding clusterers ranked by internal measure "silhouette coefficient" 98
6.12 Impact of MaCC by incrementally adding clusterers ranked by external measure "V-measure" 99
6.13 Impact of MiMaCC 100

7.1 Impact of various resources and rules in the task of name variation detection . . 107
7.2 Error distribution by categories (Type I error: missing variants, Type II error: wrong variants) 109
7.3 Type I long tail effect 122
7.4 Type II long tail effect 123
7.5 Performance of 21 clustering algorithms for all names (known prior K) 124
7.6 Performance of 21 clustering algorithms for person names (known prior K) . . 124
7.7 Performance of 21 clustering algorithms for organization names (known prior K) 124
7.8 Performance of 21 clustering algorithms for GPE names (known prior K) . . . 125
7.9 Performance of 21 clustering algorithms for all names (unknown prior K) . . . 125
7.10 Performance of 21 clustering algorithms for person names (unknown prior K) . 125
7.11 Performance of 21 clustering algorithms for organization names (unknown prior K) 126
7.12 Performance of 21 clustering algorithms for GPE names (unknown prior K) . . 126

7.13 An example for comparison of clustering algorithms 131

7.14 Impact of MiCC (known K) 133

7.15 Impact of MiCC (unknown K) 133

7.16 Four combination schemes of MaCC (known K) 137

7.17 Four combination schemes of MaCC (unknown K) 139

8.1 A motivating example of searching collaborative instances. 150

Chapter 1

Introduction

Our human society is a connected network consisting of individuals. We are leveraging others' expertise from time to time. For example, before we make important financial, shopping and medical decisions, we often ask opinions from our family members, friends, professionals and experts. The motivation of doing so is to gain more confidence in making a right decision by evaluating, weighing opinions from various people and combining them through our own thinking process. We not only benefit from others' intelligence, but also contribute ourselves by collaborating with others in a team or group. Our society can make progress not because of any individual's success, but because of the common efforts from many people.

The collaborative idea of leveraging "crowd intelligence" has been explored in Natural Language Processing (NLP) and machine learning. For example, many NLP (e.g., information extraction) techniques have gone beyond the limitation of single document and now start taking advantages of redundant and complementary information contained in multiple documents. With the explosive increase of data on the web, the same facts no longer exist on their own and they are expressed in different forms in multiple topically-related documents. In machine learning, ensemble-based approaches have been proposed for classification, clustering and ranking problems [38, 76, 85, 91]. For example, co-training [5] leverages two collaborative classifiers each of which is learned from a single view of the data.

In this thesis, we present two collaborative learning schemes, namely, "Collaborative Ranking" (CR) for ranking problems, and "Collaborative Clustering" (CC) for clustering problems.

We briefly describe the basic ideas of the two collaborative learning schemes in section 1.1 and section 1.2 respectively, by answering the following questions: (1) what are the intuitions behind the ideas? (2) what are exactly the ideas?

1.1 Collaborative Ranking

A ranking problem is typically formulated as ranking a collection of candidate "objects" with respect to a "query". The relative order of objects in the list can represent the likelihood of relevance, preference, depending on applications. In information retrieval, a "query" is typically a string while a candidate "object" is a web page. In this thesis, we have generalized the meanings of query and candidate object which can be varied in different natural language processing tasks. For example, parsing can be formulated as ranking multiple possible parsing trees with respect to a sentence (query) [10, 24, 46], machine translation can be formulated as ranking multiple translation hypotheses (objects) with respect to a source sentence (query) [70, 81], anaphora resolution can be formulated as ranking multiple antecedents with respect to an anaphora (query) [95], question answering can be formulated as ranking multiple possible answers (objects) with respect to a question (query) [74], guided summarization can be formulated as ranking multiple sentences (objects) from various topically related documents with respect to a topic (query)[1]. If not specified, the query and candidate object discussed in this thesis are "generalized query" and "generalized candidate object".

In order to illustrate our collaborative ranking idea, let us consider the following scenario:

> Mr. Turner desires to buy a camera so that he can take pictures when he travels. A rational way to help him make a decision is to make a check list such that he can evaluate all the important features of a camera, for example, brand, production date, weight, size, max resolution, optical zoom ability, LCD screen size, battery life, storage size etc. Unfortunately, like most of novices in photography, Mr. Turner does not know much about those technical specifications except some basic ones. Therefore one reasonable way to help him is to focus on some most important

[1] http://www.nist.gov/tac/2011/Summarization/Guided-Summ.2011.guidelines.html

features he can understand and for each feature, an autonomous agent can present him a ranking of cameras. For example, by the feature of price, an agent can show him a ranking from the cheapest to the most expensive; by the feature of production date, an agent can show him a ranking from the latest model to the oldest; by the feature of customer review, an agent can let him know a ranking from five star to one star. Finally, given all those rankings, he needs to make a compromise and choose the one which best fits him. To conclude, the final decision is reached by coordinating multiple rankings from various collaborative agents. An alternative way is to find out some customers who are most similar to Mr. Turner from various aspects, e.g., age, budget of buying a camera. The solution is simple, Mr. Turner can be recommended with a camera which has been bought most by those customers who are similar to Mr. Turner.

In the above scenario, we see two types of collaborations that lead to a final decision. The first type of collaboration comes from multiple autonomous agents which provide multiple rankings, and the second type of collaboration comes from multiple customers that are similar to Mr. Turner.

Similarly, our collaborative ranking scheme aims to seek collaborations from two levels:

(1) query-level: search a group of query collaborators, and make the joint decision from the group containing the collaborators and the query. This can be compared to the second type of collaboration in the example in which those customers who have similar preferences as Mr. Turner can be considered as his query collaborators.

(2) ranker-level: design a group of multiple rankers, and make the joint decision from multiple rankings. This can be compared to the first type of collaboration in the above example in which each autonomous agent can be considered as a ranker.

Figure 1.1 presents an intuitive illustration of 5 ranking approaches, including traditional non-collaborative ranking, and 4 collaborative ranking forms: micro collaborative ranking (MiCR), macro collaborative ranking (MaCR), micro-macro collaborative ranking (MiMaCR) and macro-micro collaborative ranking (MaMiCR).

Compared with traditional non-collaborative ranking that only leverages the information

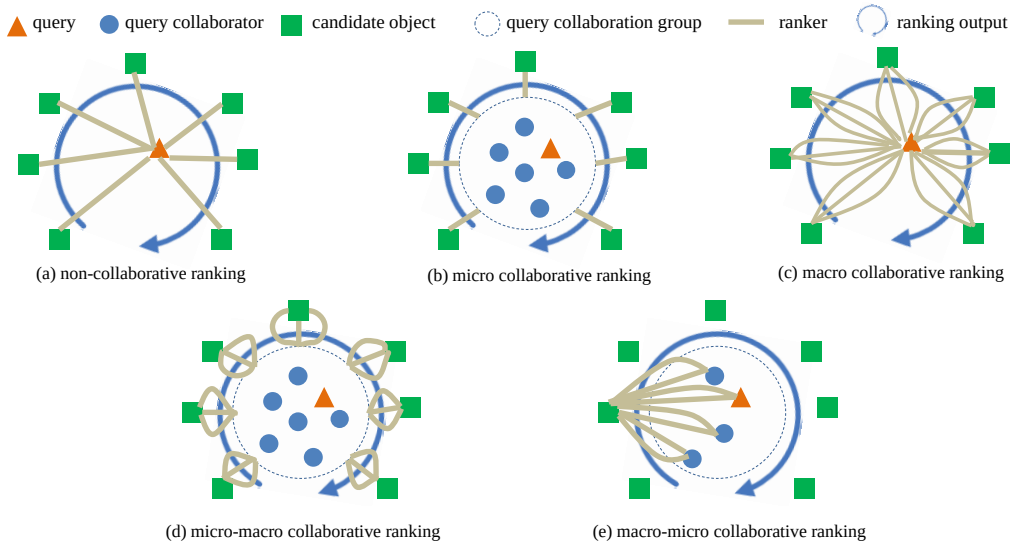


Figure 1.1: Non-collaborative ranking and four collaborative ranking approaches.

contained in the query and only applies one ranking function (Figure 1.1 (a)), the 4 collaborative ranking approaches have the following advantages:

(1) MiCR (corresponding to query-level[1] collaboration) leverages the information contained in the collaborators of a query. Figure 1.1 (b) demonstrates that 6 query collaborators together with the query form a query collaboration group and the ranking score for each candidate object is computed based on the query collaboration group using a singleton ranker.

(2) MaCR (corresponding to ranker-level[2] collaboration) integrates the strengths from two or more rankers. Figure 1.1 (c) demonstrates an example of 3 rankers and the ranking score for each candidate object is computed based on the isolated query using 3 rankers.

(3) MiMaCR combines the advantages from MiCR and MaCR as shown in Figure 1.1 (d). The ranking score for each candidate object is computed based on the query collaboration group using 3 rankers.

(4) MaMiCR also combines the advantages from MaCR and MiCR as shown in Figure 1.1 (e), but it differs from MiMaCR in that MiMaCR applies multiple rankers on the query collaboration group, while MaMiCR applies multiple rankers on each member in the query collaboration group.

We make the following hypotheses:

(1) A ranking problem can benefit from redundant and/or complementary information con-

[1]Query is normally expressed by a small-scale data structure, so we call it micro.
[2]Ranker is normally implemented by a large-scale algorithm, so we call it macro.

tained in query collaborators. The meaning of "query collaborator" varies depending on applications, e.g., in document retrieval, a query collaborator can be a name string which is syntactically or semantically similar with the given query; in entity linking, a query collaborator is a relevant document that contains the same name string as the query.

(2) A ranking problem can benefit from diversified ranker collaborators. The diversity of rankers can be implemented via exploring different ranking algorithms, training with different data and sampling strategies, developing different training models, etc.

(3) A ranking problem can further benefit from the combination of query collaborators and ranker collaborators.

We validated the above hypotheses through a case study of entity linking task defined in the Knowledge Base Population (KBP) track [50] at Text Analysis Conference (TAC). The goal of entity linking is to rank a list of candidate Knowledge Base (KB) entries according to their linking(coreference) likelihood to a query. Each query is associated with a name and its context document.

In the example shown in Figure 1.2, the query name is "NASA" and it is associated with a context document. A typical entity linking system goes through two steps, in the first step, it retrieves 7 possible KB entries in the knowledge base (such as Wikipedia) each of which refers to an unambiguous meaning of "NASA"; in the second step, it ranks the 7 KB entries using a known ranking algorithm, and returns kb1 as the correct answer. For query level collaboration (MiCR), it first retrieves more documents that mention "NASA", and by document clustering, it uses the documents that are clustered together with the query context document as query collaborators. Then a ranking algorithm can be applied on the query collaboration group rather than the single query. We hypothesize that those extra documents can provide redundant and complementary information that can benefit ranking. For ranker-level collaboration (MaCR), multiple rankings can be obtained through diverse ranking models and then can be combined to obtain a new ranking. We hypothesize that the combined ranking result can outperform the individual ranking result because it leverages the advantages in multiple ranking models. For either type of collaboration or their combination MiMaCR, the goal is to enhance the ranking score of *kb1* so that it can be output as the correct answer.

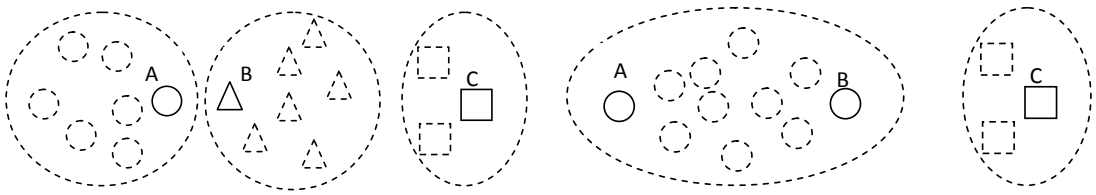
Figure 1.2: An example of entity linking task.

1.2 Collaborative Clustering

Clustering is another important problem in machine learning which is formulated as organizing a set of instances into groups such that members in each group share some similarity while members across groups are dissimilar. The meaning of instance varies depending on applications. For example, document clustering aims to organize documents (instances) into a predetermined or automatically derived number of groups so that the documents in each group share the same topic and the documents in different groups are topically dissimilar. Within-document coreference resolution aims to group entity mentions into clusters so that all the mentions in a cluster refer to the same entity. Web people search aims to group web pages that contain the same person name into clusters so that in each cluster, all the web pages refer to the same person entity. A more general clustering problem that extends web people search aims to group documents that contain a mention name or its variants into clusters so that in each cluster, all the documents refer to the same name entity.

Clustering problem is known to suffer from little prior knowledge of the underlying data population. Particularly, if the number of instances to be clustered is relatively small, it is hard to predict the actual structure in those instances. In this case, it will be desirable to find collaborative instances and add them into the original data set. Thus, the clustering algorithm can take the advantage of more interactions among the instances and help to better discover the inherent structure in the origin data set. In one example shown in Figure 1.3 (a), the original data set contains three instances A, B and C, since A and B are closer measured by some dis-

tance function, it is likely to be clustered together by some clustering algorithm. However, the actual case is that they are located at the boundaries of two different clusters. By adding more instances, it is easier for the clustering algorithm to reshape the two distinguishable clusters. In the other example shown in Figure 1.3 (b), the original data set also contains three instances A, B and C. Since B and C are closer, it is likely that some clustering algorithm clusters them together. However, the actual case is that A and B are boundary instances of the same cluster. By adding more instances, it is easier for the clustering algorithm to recognize that A and B belong to the same cluster.



(a) (b)

Figure 1.3: Two motivating examples for instance-level collaborative clustering

It is also known that every proposed clustering algorithm has its strengths and weaknesses. For example, the well-known K-Means clustering is sensitive to initial settings and thus might get stuck in a local minimum. Furthermore, the effectiveness of clustering algorithm also depends on distance or similarity measure, however, finding such measures is not trivial, especially in high-dimensional space (typically true for many NLP data sets). It will be desirable to apply a collection of diversified clustering approaches (either by varying the clustering algorithm or varying the similarity measure) and then produce a final clustering by combining the various clustering results. An example is shown in Figure 1.4.

Based on the above motivations, we present a collaborative clustering scheme for clustering problems.

(1) MiCC (corresponding to instance-level[1] collaboration) leverages the information contained in the instance collaborators. Good examples are shown in Figure 1.3 (a) and Figure 1.3 (b). However, if too many noisy collaborators are added, the structure of underlying data can also be destroyed.

[1]Instance is normally expressed by a small-scale data structure, so we call it micro.

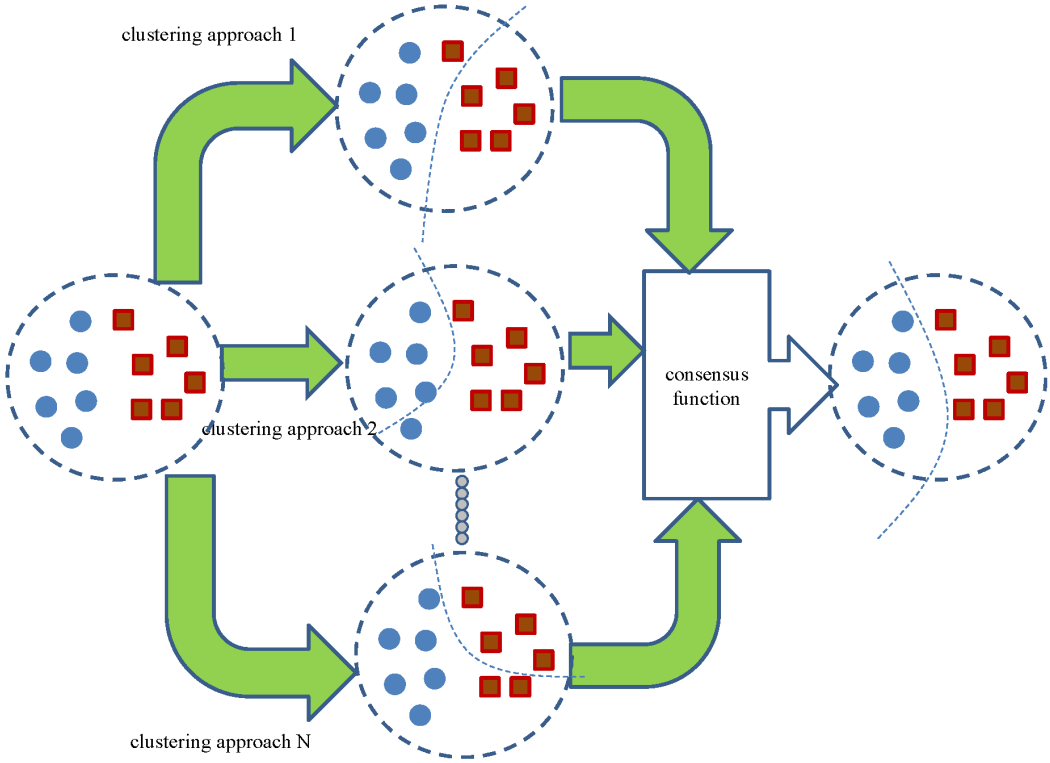


Figure 1.4: An example of clusterer-level collaborative clustering.

(2) MaCC (corresponding to clusterer-level[1] collaboration) integrates the strengths from two or more clustering algorithms, as shown in Figure 1.4.

(3) MiMaCC combines the advantages from MiCC and MaCC.

We make the following hypotheses:

(1) A clustering problem can benefit from redundant and/or complementary information contained in instance collaborators. The meaning of "instance collaborator" varies depending on applications, e.g., in web people search, each instance is associated with a person name and its context web page, we can find "instance collaborators" by retrieving more web pages that contain the name.

(2) A clustering problem can benefit from diversified clusterer collaborators. The diversity of clusterings can be implemented via exploring different clustering algorithms, developing various similarity measures to compute similarity of two instances, etc.

(3) A clustering problem can further benefit from the combination of instance collaborators and clusterer collaborators.

We validated the above hypotheses through two NLP related case studies, one is document

[1]Clusterer is normally implemented by a large-scale algorithm, so we call it macro.

clustering (Chapter 6) and the other is name entity clustering (Chapter 7).

In document clustering, the goal is to cluster documents into groups such that documents in one cluster are more similar to each other than the documents assigned to different clusters, for example, the documents in each cluster focus on one specific topic, business, politics, technology etc. For instance-level collaboration, the system retrieves more documents and applies a clustering algorithm on the expanded set. For clusterer-level collaboration, it assembles multiple clusterings generated from diversified clusterers and generates the final clustering by applying a consensus function.

In name entity clustering, the goal is to cluster instances (each instance is associated with a mention name and a context document) so that instances in one cluster refer to the same entity. There are two challenges for name entity clustering, one is name variation (an entity can have multiple mention names, therefore multiple instances with different names could actually be in the same cluster), and the other is name disambiguation (a mention name can refer to multiple entities, therefore multiple instances with the same name may actually be in multiple clusters). In the example shown in Figure 1.5, a good system should not only detect name variants of "Michael Jeffrey Jordan" in document D2 and "Michael Jordan" in document D3, but also disambiguate the "Michael Jordan" in document D1 from the "Michael Jordan" in document D2. For instance-level collaboration, the system retrieves more documents that contain "Michael Jordan". The clustering of the original set of instances can be obtained through clustering on the expanded set of instances. For clusterer-level collaboration, it combines multiple clusterings and generates the final clustering based on a consensus function.

1.3 Terminology

- query: The meaning of query varies according to applications. For example, in the most familiar information retrieval setting, it means a string; in entity linking, a query is a complicated object which consists of a name string and a context doc id which links to a concrete document. In the former case, a query can be stored in a variable using some programming language, while in the later case, a query has to be stored in a class or

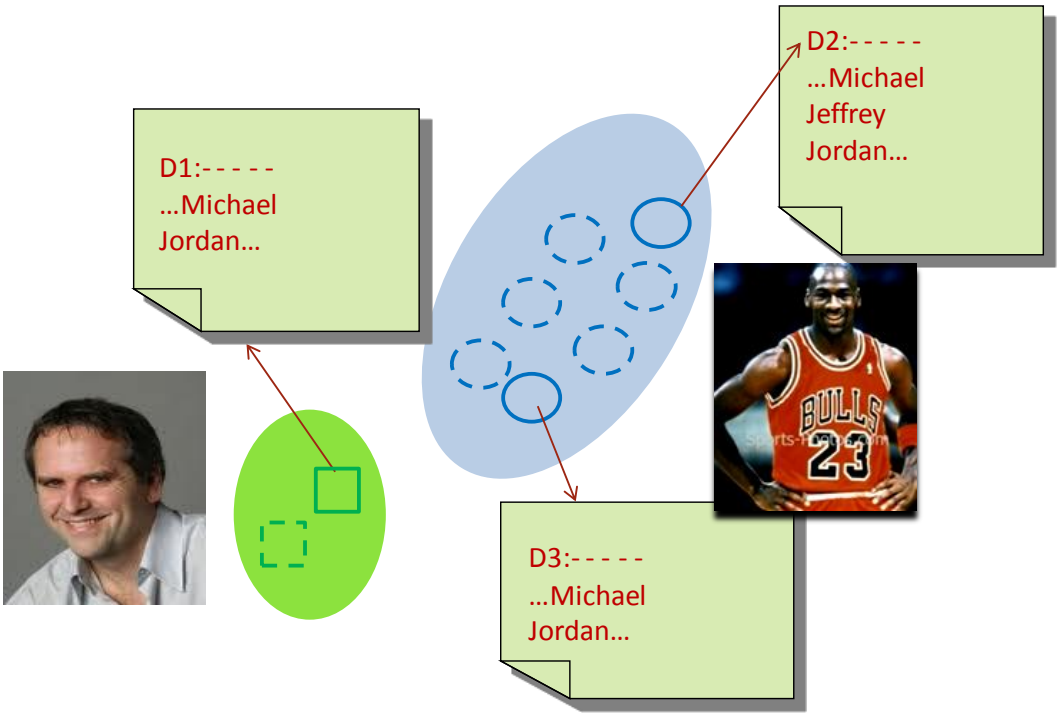


Figure 1.5: An example of name entity clustering.

structure using some programming language. In this thesis, we do not give a general meaning to query and let practitioner assign its meaning in different ranking problems.

- instance: in this thesis, instance is mentioned in the context of clustering problem in which multiple instances form the dataset to be clustered. An instance is also called a pattern, usually represented as a vector of measurements, or a point in a multi-dimensional space.

- collaborator: in this thesis, a collaborator refers to either a query collaborator or an instance collaborator. A query collaborator should have the same data structure as the query itself. For example, if query is a string, then its query collaborators should also be strings which are different from the string in the query. For a more concrete example, if the query is "Barack Obama", then its query collaborators could be other strings related with the query, e.g., "white house". In entity linking, a query collaborator has the same data structure as the query both of which have the same name but two different context doc ids. An instance collaborator should also have the data structure as the instance. For example, in document clustering, all the instances are documents, then an instance collaborator should also be a document. For name entity clustering, an instance is a struc-

ture consisting of name and context doc id, therefore an instance collaborator should also have the same structure, i.e., consisting of the same name but a different context doc id.

- ranker: a ranker is a ranking model which produces rankings of candidate objects in a ranking problem.
- clusterer: a clusterer is a clustering approach which produces clustering of the dataset in a clustering problem.

1.4 Thesis Structure

This thesis is mainly organized into two major parts, namely, collaborative ranking and collaborative clustering.

- In Chapter 2, we survey relevant literature in ranking and clustering. Given a huge volume of literature in ranking and clustering, we will not cover every aspect of ranking and clustering, instead, we briefly state the core problem, main techniques that have been proposed, and most importantly, related research that has greatly motivated the main topic (collaborative ranking and collaborative clustering) presented in this thesis.
- In Chapter 3, we present the theory of "collaborative ranking". We propose four specific forms of collaborative ranking: micro (query level) collaborative ranking (MiCR), macro (ranker level) collaborative ranking (MaCR), micro-macro collaborative ranking (MiMaCR) and macro-micro collaborative ranking (MaMiCR). We present the algorithm for each of the four collaborative ranking approaches.
- In Chapter 4, we take entity linking as a case study of collaborative ranking. Entity linking involves two important steps: (1) generation of candidate KB entries (2) ranking of candidate KB entries. In this thesis, we focus on the later step and extensively study the performance of 8 baseline ranking models. We then apply MiCR, MaCR and MiMaCR algorithms and validate their effectiveness.
- In Chapter 5, we present the theory of "collaborative clustering" which involves micro (instance level) collaborative clustering (MiCC), macro (clusterer level) collaborative

clustering (MaCC) and micro-macro collaborative clustering (MiMaCC). A key issue in MiCC is to find a good internal measure such that the selected instance collaborators can help optimize (or approximately optimize) the structure in the original set of instances. Therefore, we extensively study 12 internal measures that we believe can cover the typical internal measures proposed so far. For MaCC, we present diverse ways to produce individual clustering methods, and we also extensively study different formulations of consensus functions.

- In Chapter 6, we take document clustering as a case study of collaborative clustering. We extensively study various factors that are involved in document clustering, including the selection of external measure, similarity function, baseline clustering algorithms, selection of internal measure, impact of MiCC, MaCC and MiMaCC.

- In Chapter 7, we take name entity clustering as another case study of collaborative clustering. We first study two classification problems, one deals with name variation identification, and the other deals with name disambiguation classification. We then elaborate the name entity clustering problem, including the study of baseline clustering algorithms, impact of MiCC and MaCC.

- In Chapter 8 we summarize our thesis and present some future directions of this thesis.

1.5 Related Publications

Some of the thesis work has been published in the following papers.

- Collaborative ranking: [18]

- Entity linking: [9, 18, 20]

- Literature survey of graph clustering: [13, 17]

- Event coreference resolution and event extraction: [14, 15, 16, 19, 49]

Chapter 2

Preliminaries

2.1 Ranking

2.1.1 Problem Formulation

In this thesis, the goal of ranking problem is to return a ranked list of objects ordered by relevance or preference or some relationship with respect to a query. Document retrieval is a typical ranking problem we study here, namely, given a query string submitted by a user, the information retrieval system returns a ranked list of documents in the descending order of relevance to the query. In our case study discussed in Chapter 4, a query consists of a name and a context document in which the name occurs, our entity linking system returns a ranked list of knowledge base (KB) entries in the descending order of coreference likelihood to the query and determines whether the top KB entry (entity) is actually what the query refers to.

Formally, let q denote a query.

Let $o^{(q)} = \left\{o_1^{(q)}, \ldots, o_{n^{(q)}}^{(q)}\right\}$ denote the candidate object set associated with q, where $n^{(q)}$ denotes the size of the $o^{(q)}$, $o_j^{(q)}$ denotes the jth object in the set.

Let $R = \{r_1, ..., r_l\}$ denote the rating labels where l denotes the number of ratings. A rating label could be represented using a string, for example, in document retrieval, a retrieved document can be rated as relevant, partially relevant, and irrelevant; it could be represented using a cardinal, e.g., 1, 2, which indicates the ranking position in the sequence.

The goal of our ranking problem is to seek a ranking function such that it produces ratings

for the set of objects, i.e., $y^{(q)} = f(o^{(q)}) = \left\{y_1^{(q)}, \ldots, y_{n^{(q)}}^{(q)}\right\}$ in which each element $y_j^{(q)}$ denotes the rating label for the object $o_j^{(q)}$, i.e.,$y_j^{(q)} \in R$.

Let $\prec$ be a preference operator which can be applied on any pair of ranking labels, i.e., $r_i \prec r_j$ indicates that an object with the rating label r_j should be ranked higher than an object with the rating label of r_i. As an example in document retrieval, the rating set consists of three labels, i.e., $R = \{$*irrelevant, partial relevant, relevant*$\}$, documents with the label of *relevant* should be ranked higher than those with the label of *partial relevant* and then higher than those with the label of *irrelevant*.

Without loss of genericity, we assume that the rating labels in R are ordered such that $r_1 \prec r_2 \prec \ldots \prec r_l$.

Learning to rank: a supervised setting

A supervised setting for ranking problem is formulated as follows: given a set of queries and each query is associated with a set of objects and corresponding rating labels (used as our training data), the goal of ranking is to learn an optimal ranking function that minimizes a certain loss function based on the training data.

Formally, let $\mathcal{Q} = \{q_1, \ldots, q_N\}$ denote the set of N queries in the training data, each query q_i is associated with a set of objects $o^{(q_i)} = \left\{o_1^{(q_i)}, \ldots, o_{n^{(q_i)}}^{(q_i)}\right\}$ and a set of ground-truth rating labels $y^{(q_i)} = \left\{y_1^{(q_i)}, \ldots, y_{n^{(q_i)}}^{(q_i)}\right\}$. Let $x_j^{(q_i)} = \varphi(q_i, o_j^{(q_i)})$ denote a feature vector associated with each query-object pair $(q_i, o_j^{(q_i)})$.

2.1.2 Non-collaborative Ranking Approaches

Earlier studies on non-collaborative ranking mainly explored unsupervised methods, e.g., vector space model such as BM25 model [75], link based algorithm such as HITS [56] and PageRank [71]. Unsupervised approaches are based on well-established statistical and probability theory, nevertheless, they suffer from some drawbacks, for example, it is hard to tune parameters. Recently, supervised approaches (named "learning to rank") that automatically learn ranking functions from training data become the focus of ranking research. In the literature, supervised approaches are categorized into three classes, namely, pointwise, pairwise, and listwise. We summarize a comparison of the three approaches in Table 2.1.

	pointwise	**pairwise**	**listwise**
approach overview	common: 1) use training samples; 2) learn the best ranking function by minimizing a given loss function; 3) apply the ranking function at ranking step		
	transform ranking to regression or classification on single objects	transform ranking to classification on object pairs	ranking by learning from lists of objects
training set	$\{\left(x_j^{(q_i)}, y_j^{(q_i)}\right)\}_{j=1,\ldots,n^{(q_i)};\ i=1,\ldots,N}$	$\{\left(x_j^{(q_i)}, x_k^{(q_i)}, y\right)\}$ $j=1,\ldots,n^{(q_i)};$ $k=1,\ldots,n^{(q_i)};$ $k \neq j;\ i=1,\ldots,N$ $y = \begin{cases} +1\ if\ x_j^{(q_i)} > x_k^{(q_i)} \\ -1\ if\ x_j^{(q_i)} \leq x_k^{(q_i)} \end{cases}$	$\{(x^{(q_i)}, y^{(q_i)})\}_{i=1,\ldots,N}$
loss function	pointwise loss, e.g., square loss [11]	pairwise loss, e.g., hinge loss [97], exponential loss [4], logistic loss [60]	listwise loss, e.g., cross entropy loss [8],cosine loss [73]
pros and cons	pros: classification is well studied cons:1) only consider one object at a time ignoring relationship among objects	pros: classification is well studied cons:1) only consider pairwise orders; 2) biased towards lists with more objects	pros: fully consider relationship among objects cons: 1) less well studied in theory
selected algorithms	Discriminative model for IR [69]; McRank [59]	SVM Ranking [52]; RankBoost [37]; RankNet [6]	ListNet [8]; RankCosine [73]

Table 2.1: Comparison of pointwise, pairwise and listwise ranking approaches.

The pointwise approaches transform the ranking problem into a classification problem, and each training unit is a query-object pair. Though classification problem is well studied, it considers each query-object pair individually and the orders of objects are purely obtained by sorting the classification probabilities in descending order. The pairwise approaches are superior to pointwise approaches since they take into accounts the relative order of object pairs. However, they still do not tackle the ranking problem in a straightforward style, in contrast, listwise approaches use the whole object list (with their rating labels) as a training unit, therefore, they are superior to pairwise or pointwise approaches.

We categorize most of the past unsupervised or supervised approaches into non-collaborative approaches because they only leverage the information contained in the query, and meanwhile only exploit one single ranking function. We will discuss our collaborative ranking scheme and representative approaches in Chapter 3.

2.1.3 Query Expansion

One form of our collaborative ranking scheme leverages the strength of query collaborators, in other words, given a query, we expand the query to a group which contains members that can potentially help the ranking task. This should be distinguished from the traditional query expansion in information retrieval although they share the same goal of improving the final ranking results. In the literature of information retrieval, query expansion is a technique that reformulates a query, and as a consequence, is capable to extend the ability of a query and improve the retrieval performance. The basic assumption is that users may not use the best terms as the query, therefore, replacing terms with synonyms, re-weighting the terms in the query can all help. An extensive survey of query expansion techniques is presented in [63]. The query level collaborative ranking differs from the traditional query expansion in two aspects, firstly, we do not focus on reformulating the query, instead, we aim to search potentially helpful query collaborators (the meaning of a query collaborator varies by applications); secondly, the ranking decision is made by taking into accounts the whole collaboration group and a key issue here is how to effectively combine the strengths from multiple query collaborators.

2.1.4 Ensemble Ranking

Another form of our collaborative ranking scheme leverages the strength of ranker collaborators, in other words, a ranking decision is made by taking into accounts decisions from multiple rankers, rather than a single ranker. This topic has started attracting attentions from our community and become known as ensemble ranking, however, the work in this topic is still limited as compared to the considerable amount of research in ensemble classification [76], which is to build a predictive classification model by integrating multiple classifiers.

[45] proposed a novel semi-supervised ensemble ranking (SSER) algorithm that learns *query-dependent* weights when combining multiple rankers in document retrieval. The algorithm is formulated as an SVM-like quadratic program which can be solved efficiently by traditional SVM approaches. Their experimental results validated the effectiveness of the algorithm by comparing with other single ranking algorithm (e.g., RankBoost, Ranking SVM, ListNet, AdaRank), however, a main drawback of their algorithm is that it has to learn a dif-

ferent set of weights *online* for each query, which is intractable for every large application. Therefore, more efficient approaches should be explored to alleviate the high computational cost for online learning.

[91] proposed an interactive algorithm *iRank* in which two rankers can teach each other so as to jointly improve the ranking performance of both rankers. More specifically, one base ranker takes the most confident ranking results from the other ranker as feedback to update its own ranking results, and vise visa. This process continues iteratively until a termination condition is satisfied (e.g., the ranking results do not change any more). However, the *iRank* algorithm also has limitations:(1) The *iRank* does not explicitly indicate that the two rankers can be re-learned during the iterations because in their algorithm description, we do not see that their rankers are feeded with training data or the parameters in the rankers can be updated. From the case study (query-focused summarization) of *iRank*, we noticed that they implemented two unsupervised rankers, and only in the first iteration, the two rankers produce initial ranking scores. In the later iterations, the ranking score of each object from one ranker is updated depending on how similar it is with the top N ($N = 1$) ranked objects from the other ranker. (2) The *iRank* does not tell how the rankers get updated if the number of rankers is greater than 2. In spite of these limitations, we have been greatly motivated by their work and we will propose an extended work that takes into accounts of unlabeled data and multiple rankers.

2.2 Clustering

2.2.1 Problem Formulation

In this thesis, we mainly focus on the following hard clustering problem, given a collection of instances, our clustering algorithm organize them into clusters such that the instances in each cluster are homogeneous (similar) while the instances across different clusters are heterogeneous (dissimilar).

Formally, given a data set of n instances $X = \{X_1, X_2, ..., X_n\}$, a hard clustering π is a partition of the data set X into K disjoint clusters $\pi = \{c_1, c_2, ...c_K\}$, such that

(1) $c_i \neq \emptyset$ for $i \in \{1, ..., K\}$.

(2) $c_i \cap c_j = \emptyset$ for $i, j \in \{1, ..., K\}$ and $i \neq j$.

(3) $c_1 \cup ... \cup c_K = X$

2.2.2 Non-collaborative Clustering Approach

Clustering is an extensively researched topic and there is considerable amount of literature. Two most recent thorough surveys have been presented in [48] and [94] respectively. A typical clustering solution is as follows [48, 94]:

(1) **pattern representation** (optionally including feature extraction and/or selection): determining the number of classes (this information may not be available in many cases), the number of instances in the total collection, and the number, type, and scale of the features available to the clustering algorithm. *Feature selection* deals with selecting the most useful features in clustering while *feature extraction* deals with transforming some input features into new salient features.

(2) **similarity computation**: a distance function or a similarity function is usually defined to measure the dissimilarity (distance function) or similarity (similarity function) of two instances in the collection. Various distance functions can be used according to the setting of clustering problem, for example, Euclidean Distance. For natural language processing problems, cosine similarity based on bag-of-word model is widely used.

(3) **grouping**: exploring various clustering algorithms, for example, hierarchical clustering produces a nested series of partitions by iteratively merging or splitting clusters based on similarity.

(4) **data abstraction** (optional): extracting a simple and compact representation of generated clusters. It could be human-oriented such that representation of each cluster is easy to explain and comprehend.

(5) **clustering evaluation** (optional): evaluating how "good" is a clustering based on some internal measure (without leveraging external information of gold clustering) or external measure (a gold clustering is known).

In our previous work, we extensively surveyed the clustering problem using graph formulation [13, 17]. From the theoretical aspect, we state the methodology of graph-based clustering

using the following five-part story :

(1) **hypothesis**. The hypothesis is that there exist groups of instances in the graph such that the similar instances are assigned in the same sub-graph while the dissimilar instances are distinguished by different sub-graphs.

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

(3) **internal quality measure**. A quality measure is proposed to answer the question: what exactly is an optimal clustering in the graph?

(4) **algorithm**. An algorithm is developed to exactly or approximately optimize the quality measure.

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

Chapter 3

Collaborative Ranking

3.1 Micro Collaborative Ranking(MiCR)

Micro collaborative ranking is characterized by integrating joint strengths from multiple query collaborators and the query itself. It is based on the following assumptions:

• Expandability: Query is expandable, that is, it is able to find potential collaborators.

• Redundancy: Collaborators and query may share redundant information.

• Diversity: Collaborators exhibit multifaceted information that may complement the information contained in the query.

• Robustness: Noisy collaborators are allowable, and they could be put under control. For example, query collaborators can be ranked and lower ranked collaborators will be removed.

Let $c^{(q)} = \{c_1, \ldots, c_k\}$ be k collaborators of a query q. For each object $o_j^{(q)}$ associated with q, we form $k+1$ feature vectors $x_j^{(q)} = \varphi(q, o_j^{(q)}), x_j^{(c_1)} = \varphi(c_1, o_j^{(c_1)}), \ldots,$
$x_j^{(c_k)} = \varphi(c_k, o_j^{(c_k)})$. Let f be a ranking function which is obtained by either an unsupervised or a supervised approach. There are two important steps that distinguish MiCR from traditional non-collaborative ranking approaches:

• Step (1): searching the best k collaborators of q.

• Step (2): computing the interaction of k collaborators at the ranking step.

Solutions for step (1) can vary from case to case. In our case study of entity linking, we transform the collaborator searching problem into a clustering problem. Collaborators of a

query are then formed by members (excluding the query) in a cluster which contains the query and k is the size of the cluster minus one.

We transform the problem of step (2) into solving a composite function g_1 such that a ranking score $y_j^{(q)}$ can be computed for each object $o_j^{(q)}$. Firstly we compute the ranking scores of collaborators and query using the ranking function f and then combine those ranking scores in some way (Formula 3.1). Another approach is to learn a supervised ranking function $f^{'}$ which takes collaborators and query as input (Formula 3.2).

$$y_j^{(q)} = g_1(f(x_j^{(q)}), f(x_j^{(c_1)}), \ldots, f(x_j^{(c_k)})) \tag{3.1}$$

$$y_j^{(q)} = g_1(\cdot) = f^{'}(x_j^{(q)}, x_j^{(c_1)}, \ldots, x_j^{(c_k)}) \tag{3.2}$$

We present three specific forms of g_1 in Formula 3.1, namely, max, min, and weighted. We can also define a special case of weighted, called "average" in which $w_0 = w_1 = \ldots = w_k = 1/(k+1)$.

- max: $y_j^{(q)} = \max(f(x_j^{(q)}), f(x_j^{(c_1)}), \ldots, f(x_j^{(c_k)}))$
- min: $y_j^{(q)} = \min(f(x_j^{(q)}), f(x_j^{(c_1)}), \ldots, f(x_j^{(c_k)}))$
- weighted: $y_j^{(q)} = w_0 f(x_j^{(q)}) + \sum_{i=1}^{k} w_i f(x_j^{(c_i)})$

in which $w_0, w_1, \ldots, w_k$ represent the weight of each query collaborator.

The practical meanings of max, min, weighted and average are:

(1) max: the overall ranking score for each object $o_j^{(q)}$ relies on the query collaborator with the maximum strength. The problem of "max" is that while it promotes the higher ranked (in truth) objects, it may also promote the lower ranked (in truth) objects. If the lower ranked objects are promoted more, ranking performance can be dropped.

(2) min: the overall ranking score for each object $o_j^{(q)}$ relies on the query collaborator with the minimum strength. The problem of "min" is that while it weakens the lower ranked (in truth) objects, it may also weaken the lower ranked (in truth) objects. If the higher ranked objects are weakened more, ranking performance can also be dropped.

(3) weight: the overall ranking score for each object $o_j^{(q)}$ relies on every query collaborator according to its strength (weight) in the group.

(4) average: the overall ranking score for each object $o_j^{(q)}$ relies on every query collaborator without distinguishing.

We will discuss three supervised versions of g_1 (Formula 3.2) in section 4.5.

A general algorithm for MiCR is presented in Algorithm 1.

Algorithm 1 MiCR Algorithm.

Input:
a query q; a set of objects $o^{(q)}$; a composite function g_1
Output:
a set of ranking scores $y^{(q)}$
1: Search k collaborators of q: $c^{(q)} = \{c_1, \ldots, c_k\}$.
2: **for** $j = 1 \rightarrow n^{(q)}$ **do**
3: Form $k+1$ feature vectors: $x_j^{(q)}, x_j^{(c_1)}, \ldots, x_j^{(c_k)}$.
4: Compute composite function: $y_j^{(q)} = g_1(\cdot)$.
5: **end for**
6: **return** $y^{(q)}$

3.2 Macro Collaborative Ranking(MaCR)

Macro collaborative ranking is featured by integrating joint strengths from multiple rankers. It is based on the following assumptions:

• Independence: Each ranker can make ranking decisions and is not interfered by any other rankers.

• Diversity: Each ranker has its own strengths in making ranking decisions.

• Collaboration: Rankers in the group could collaborate to make a consensus decision under some mechanism.

Let $x_j^{(q)} = \varphi(q, o_j^{(q)})$ be the feature vector formed from the pair consisting of query q and an associated object $o_j^{(q)}$. Let $\mathcal{F}^* = \{f_1, \ldots, f_m\}$ be the m existing ranking functions. We transform the simulation of collaboration among rankers into computing the following composite function g_2:

$$y_j^{(q)} = g_2(f_1(x_j^{(q)}), \ldots, f_m(x_j^{(q)})) \tag{3.3}$$

Similar with MiCR, g_2 can be expressed by max, min, weighted (average) respectively:

- max: $y_j^{(q)} = \max\{f_i(x_j^{(q)})\}_{i=1}^m$
- min: $y_j^{(q)} = \min\{f_i(x_j^{(q)})\}_{i=1}^m$
- weighted: $y_j^{(q)} = \sum_{i=1}^m w_i f_i(x_j^{(q)})$

It is worth noting that max and min can be useful only if the ranking scores produced by various rankers can be compared to each other directly, which means that the ranking scores from different rankers should be normalized.

A special form of ranking problem is that only the best object is required as output. In this case, we have another version of g_2 which is called voting:

- voting: $y_j^{(q)} = \sum_{i=1}^m sig(f_i(x_j^{(q)}))$

in which $sig(\cdot)$ is an indicator function

$$sig(\cdot) = \begin{cases} 1 & \text{if } f_i \text{ outputs } o_j^{(q)} \text{ as the best object} \\ 0 & \text{otherwise} \end{cases}$$

A general algorithm for MaCR is presented in Algorithm 2.

Algorithm 2 MaCR Algorithm.

Input:
a query q; a set of objects $o^{(q)}$; a set of m ranking functions $\mathcal{F}^*$; a composite function g_2
Output:
a set of ranking scores $y^{(q)}$
1: **for** $j = 1 \rightarrow n^{(q)}$ **do**
2: Form a feature vector $x_j^{(q)}$.
3: Compute ranking scores: $f_1(x_j^{(q)}), \ldots, f_m(x_j^{(q)})$.
4: Compute composite function: $y_j^{(q)} = g_2(\cdot)$.
5: **end for**
6: **return** $y^{(q)}$

3.3 Micro-Macro Collaborative Ranking (MiMaCR)

The above two ranking approaches can be further integrated into a joint model which is named Micro-Macro Collaborative Ranking (MiMaCR). In order to simulate the query-level collaboration and ranker-level collaboration jointly, we compute the following complex composite function g_3:

$$y_j^{(q)} = g_2(g_1(\cdot)) \tag{3.4}$$

in which, for each object $o_j^{(q)}$, firstly we compute m micro-ranking scores using m ranking functions on the query-level collaborators

$$m \begin{cases} g_1(f_1(x_j^{(q)}), f_1(x_j^{(c_1)}), \ldots, f_1(x_j^{(c_k)})) \\ \ldots \\ g_1(f_m(x_j^{(q)}), f_m(x_j^{(c_1)}), \ldots, f_m(x_j^{(c_k)})) \end{cases}$$

and secondly, we compute a macro-ranking score using formula 3.3.

We can similarly define g_1 and g_2 as those in MiCR and MaCR. A general algorithm for MiMaCR is presented in Algorithm 3.

Algorithm 3 MiMaCR Algorithm.

Input:
a query q; a set of objects $o^{(q)}$; a set of ranking functions $\mathcal{F}^*$; composite functions g_1, g_2
Output:
a set of ranking scores $y^{(q)}$

```
Search k collaborators of q: c^(q) = {c_1, ..., c_k}.
for j = 1 → n^(q) do
    Form k + 1 feature vectors: x_j^(q), x_j^(c_1), ..., x_j^(c_k).
    Compute m micro-ranking scores using F* and g_1.
    Compute the macro-ranking score using g_2.
end for
return y^(q)
```

3.4 Macro-Micro Collaborative Ranking (MaMiCR)

MaMiCR differs from MiMaCR in that MiMaCR applies multiple rankers on the query collaboration group, while MaMiCR applies multiple rankers on each member in the query collaboration group.

we compute the following complex composite function g_4:

$$y_j^{(q)} = g_1(\underbrace{g_2(\cdot), \ldots, g_2(\cdot)}_{k+1}) \tag{3.5}$$

in which, for each object $o_j^{(q)}$, we compute $k+1$ macro-ranking scores

$$g_2(f_1(x_j^{(q)}), \ldots, f_m(x_j^{(q)}))$$
$$g_2(f_1(x_j^{(c_1)}), \ldots, f_m(x_j^{(c_1)}))$$
$$\ldots$$
$$g_2(f_1(x_j^{(c_k)}), \ldots, f_m(x_j^{(c_k)}))$$

for feature vectors $x_j^{(q)}, x_j^{(c_1)}, \ldots, x_j^{(c_k)}$ respectively, and secondly, we compute a micro-ranking score using formula 3.1.

We can similarly define g_1 and g_2 as those in MiCR and MaCR. A general algorithm for MaMiCR is presented in Algorithm 4.

Algorithm 4 MaMiCR Algorithm.

Input:
a query q;
a set of objects $o^{(q)} = \left\{o_1^{(q)}, \ldots, o_{n^{(q)}}^{(q)}\right\}$;
a set of ranking functions $\mathcal{F}^* = \{f_1, \ldots, f_m\}$;
composite functions g_1, g_2

Output:
a set of ranking scores $y^{(q)} = \left\{y_1^{(q)}, \ldots, y_{n^{(q)}}^{(q)}\right\}$

1: Search k collaborators of q: $c^{(q)} = \{c_1, \ldots, c_k\}$.
2: **for** $j = 1 \to n^{(q)}$ **do**
3: Form $k+1$ feature vectors: $x_j^{(q)}, x_j^{(c_1)}, \ldots, x_j^{(c_k)}$.
4: Compute $k+1$ macro-ranking scores.
5: Compute the micro-ranking score for $y_j^{(q)}$.
6: **end for**
7: **return** $y^{(q)}$

3.5 Summary

In this chapter, we propose four specific forms of collaborative ranking, namely, MiCR, MaCR, MiMaCR, and MaMiCR. We present a detailed algorithm for each of the schemes, and in next chapter, we will validate whether these proposed schemes are effective in a real application of entity linking.

Chapter 4

Entity Linking: A Case Study of Collaborative Ranking

4.1 Problem Formulation

The entity linking task aims to align a textual mention of a named entity (person, organization or geo-political) to an appropriate entry in a knowledge base (KB), which may or may not contain the entity. More formally, given a large corpus $\mathcal{C}$, let $q = (q.id, q.string, q.text)$ denote a query in the task which is a triple consisting of query id ($q.id$), name string ($q.string$) and context document ($q.text \in \mathcal{C}$). Let $o^{(q)} = \left\{o_1^{(q)}, \ldots, o_{n^{(q)}}^{(q)}\right\}$ denote the candidate KB entries associated with the query. Each KB entry is a tuple consisting of KB id, KB title, KB infobox (attribute-value pairs) and KB text. The goal is to rank the KB entries and determine whether the top entry id should be considered as the answer, otherwise NIL should be returned.

4.2 Related Research

There has been an increasing amount of research on entity linking, especially through the evaluations of KBP2009 and KBP2010. Various unsupervised or supervised approaches have been proposed, as summarized in [50, 65]. However, most of the previous research mainly focused on individual ranking algorithms and feature contributions of supervised ranking models, for example, [26] applied Bag-of-Word (BOW) unsupervised ranking approach, [31] applied a

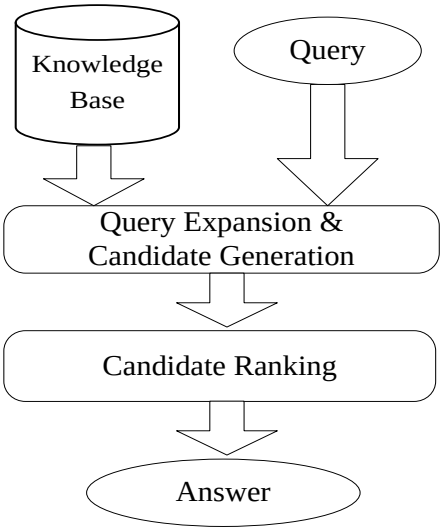


Figure 4.1: Framework of entity linking system.

pairwise ranking appraoch *SVM Rank model*, and [98] applied a pointwise ranking approach *SVM classifier*. [102] compared the listwise ranking approach *ListNet*, pairwise ranking approach *Ranking Perceptron*, and pointwise ranking approaches *SVM* and *Perceptron*. Their experimental results show that the listwise approach can outperform pairwise or pointwise approaches. [44] proposed a entity-mention model which takes the product form of 3 separate models, i.e., entity popularity model, entity name model, and entity context model. As far as we know, our work is the first to explore query-level collaboration and ranker-level collaboration for entity linking problem.

4.3 General Framework

A general framework of entity linking consists of two crucial components as shown in Figure 4.1, one for candidate generation, the other for candidate ranking. We developed the first component by following the procedures described in [20] which extensively leveraged resources mined from Wikipedia. The first component is important in that it determines the qualities of candidate KB entries. If the true KB entry is excluded from this candidate set, the second component makes no sense. If there are too many very similar KB entries with the true KB entry, it will also make the second component harder to rank them correctly. In this thesis, we focus on the ranking problem itself.

4.4 Baseline Rankers

We developed 8 baseline rankers, including 4 unsupervised rankers (f_1, f_2, f_3, f_4) and 4 supervised rankers(f_5, f_6, f_7, f_8). We use $f_1, \ldots, f_8$ to denote the 8 ranking functions.

•Naive (f_1): since the answer for each query can either be a KB id or NIL, the naive ranker simply outputs NIL for all queries.

•Entity (f_2): f_2 is defined as weighted combination of entity similarities in three types (person, organization and geo-political). Name entities are extracted from $q.text$ and KB text respectively using Stanford NER toolkit[1]. The formulas to compute entity similarities are defined in [96].

•TF-IDF (f_3): f_3 is defined as cosine similarity between $q.text$ and KB text using TF-IDF weights.

•Profile (f_4): f_4 is defined as profile similarity between $q.text$ and KB text [20]. We used a slot filling toolkit [21] to generate the profile (attribute-value pairs) for each query.

•Maxent (f_5): a pointwise ranker implemented using OpenNLP Maxent toolkit[2] which is based on maximum entropy model.

•SVM (f_6): a pointwise ranker implemented using SVM^{light} [51].

•SVM ranking (f_7): a pairwise ranker implemented using SVM^{rank} [53].

•ListNet (f_8): a listwise ranker presented in [8].

The four supervised rankers apply exactly the same set of features except that SVM ranking (f_7) needs to double expand the feature vector. The features are categorized into three levels, surface features [31, 102], document features [31, 102], and profiling features (entity attributes that are extracted by a slot filling toolkit [21]). Furthermore, as described in [102], we also added a top candidate validation module in the ListNet ranker except that we implemented the binary classifier based on maximum entropy model and used slightly different features.

[1] http://nlp.stanford.edu/software/CRF-NER.shtml
[2] http://maxent.sourceforge.net/about.html

4.5 MiCR for Entity Linking

We convert the collaborator searching problem into a clustering problem, i.e., for a given query q in the task, we retrieve at most $K = 300$ documents from the large corpus $\mathcal{C}$, each of which contains $q.string$; we then apply a clustering algorithm to generate clusters over the documents, and form query collaborators (excluding $q.text$) from the cluster that contains $q.text$.

We experimented the following two clustering approaches:

(1)*agglomerative clustering*: it iteratively merges clusters from singleton documents until a stop threshold is reached. Document similarity is defined as cosine similarity using TF-IDF weights. We applied group-average linking strategy to merge clusters [63].

(2)*graph-based clustering*: it iteratively partitions clusters from one single cluster until a stop threshold is reached. Document similarity is similarly defined as agglomerative clustering. We selected normalized spectral clustering as our clustering algorithm [82].

We first selected f_3 as our basic ranking function, and investigated whether the ranker can benefit from query collaborators formed by either agglomerative clustering or graph clustering. We implemented three versions of composite function g_1 (*max*, *min* and *average*), and experimented their performance on three unsupervised rankers f_2, f_3, f_4 respectively.

Last, we implemented three supervised versions of g_1 (Maxent, SVM and ListNet respectively) by adding cluster-level features and retraining the models in three supervised rankers f_5, f_6, f_8 respectively. Cluster-level features include maximum, minimum, average tfidf/entity similarities between the candidate and the query collaboration group.

4.6 MaCR for Entity Linking

We implemented two versions of composite function g_2, *average* and *voting*. Furthermore, we investigated how the performance can be affected by incrementally adding more rankers into the ranker set $\mathcal{F}^*$. To do so, we first sorted the 8 rankers according to their performance on the *development* set from the highest to the lowest, and starting with the highest performance ranker, we added one ranker at a time, until we have all the 8 rankers. It is worth noting that, when there are even number of rankers in the set $\mathcal{F}^*$, "ties" could take place using *voting*

function. In order to break the ties, we rank the candidate higher if it is outputted as the answer from a higher performance ranker.

4.7 MiMaCR for Entity Linking

We investigated how the final performance can be boosted by jointly computing micro-ranking scores and macro-ranking score as described in section 3.3.

4.8 Experimental Results

4.8.1 Data and Evaluation Metric

We used TAC-KBP2009 evaluation data as our training (75%) and development set (25%), and used TAC-KBP2010 evaluation data as our blind testing set (shown in Table 4.1).

Corpus	Queries			
	PER	ORG	GPE	Total
Training&Dev	627	2710	567	3904
Testing	750	750	750	2250

Table 4.1: Training, development and testing corpus.

The reference KB consists of 818,741 entries which are extracted from an October 2008 dump of English Wikipedia. The source text corpus (denoted as $\mathcal{C}$ in section 4.1) consists of 1,777,888 documents in 5 genres (mostly Newswire and Web Text).

We used the official evaluation metric for TAC-KBP2010 entity linking task, that is, micro-averaged accuracy. It is computed by

$$micro-averaged\ accuracy = \frac{\#correct\ answers}{\#queries}$$

An answer is considered as correct if the system output (either a KB entry id or NIL) exactly matches the key.

	Overall (%)			PER (%)			ORG (%)			GPE (%)		
	All	KB	NIL	All	KB	NIL	All	KB	NIL	All	KB	NIL
Naive	54.5	0.0	**100**	70.8	0.0	**100**	59.7	0	**100**	33.0	0	**100**
Entity	65.6	48.6	79.7	82.1	52.1	94.5	68.4	46.2	83.3	46.1	48.5	41.3
Tfidf	68.3	45.0	87.7	83.6	54.3	95.7	66.2	45.9	80.0	54.9	40.3	84.6
Profile	75.0	58.7	88.6	90.8	82.2	94.4	73.3	62.7	80.4	61.0	46.1	91.1
Maxent	77.4	72.3	81.6	86.5	**82.6**	94.4	73.3	62.7	80.4	61.0	**71.5**	72.1
SVM	78.1	**73.0**	82.3	91.1	81.7	94.9	78.7	**70.0**	84.6	64.4	71.1	51.0
SVM Rank	80.3	66.7	91.7	**91.3**	76.3	97.6	77.3	59.7	89.1	72.3	66.7	83.8
ListNet	**81.1**	69.7	90.6	90.8	77.6	96.2	**79.0**	64.0	89.1	**73.5**	69.7	81.4

Table 4.2: Comparison of 8 baseline rankers.

4.8.2 Performance of 8 Baseline Rankers

Table 4.2 shows the performance of the 8 baseline rankers in 4 columns: *Overall* for all queries, *PER* for person queries, *ORG* for organization queries, and *GPE* for geo-political queries. Each column is further split into *All*, *KB* (for Non-NIL queries) and *NIL* (for NIL queries). It shows that all the four supervised rankers perform better than the four unsupervised rankers. Naive ranker obtains the lowest overall micro-average accuracy (54.5%) but the highest NIL accuracy (100%). Among the four unsupervised rankers, *profile* ranker performs the best, which clearly shows that the extracted attributes of entities are effective for disambiguating confusable names. For example, our data analysis shows that the attribute value of "*per:alternative-name*" from the context document is particularly useful if a person query is only mentioned by its last name. The attribute "*per:title*" is another important indicator to discriminate one person from the other. For geo-political queries, if the query is a city name, attribute "*gpe:state*" is useful to distinguish cities with the same name but in different states or provinces. Among the four supervised rankers, ListNet outperforms SVM ranking and then SVM ranking outperforms the two pointwise rankers. It may confirm previous research findings that listwise ranking is superior to pairwise ranking and pairwise ranking is superior to pointwise ranking [8, 102]. The best baseline ranker (ListNet) obtains an absolute overall accuracy gain of 26.6% over the naive ranker.

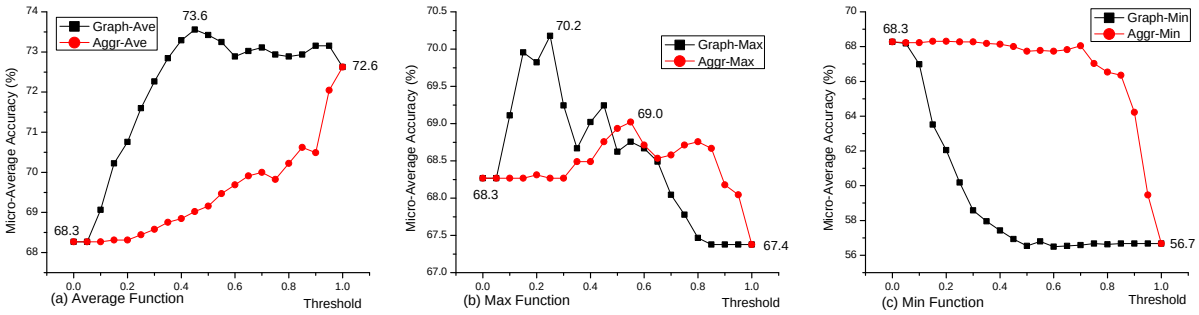


Figure 4.2: MiCR: comparison of average, max, and min functions with Graph and Agglomerative (Aggr)-based query collaborator searching strategies (*tfidf* ranker).

4.8.3 Impact of MiCR

To study the impact of MiCR, we first select f_3 (*tfidf* ranker) as our ranking function. Figure 4.2 shows the performance of applying different query collaborator searching strategies (graph or agglomerative clustering) and different versions of g_1 (average, max and min respectively). We intentionally adjust the meaning of threshold (x-axis) for both graph clustering and agglomerative clustering, such that at threshold 0, both clustering algorithms generate the largest number of clusters (i.e., each document is a cluster), and at threshold 1, they generate only one cluster. We now take the *average* function (Figure 4.2 (a)) into considerations, as graph clustering algorithm gradually partitions from one cluster (corresponding to threshold 1) to more clusters, the number of query collaborators gradually reduces, meanwhile, the accuracy gradually increases and arrives at the highest (73.6%) at threshold of 0.45, which clearly shows that removing noisy collaborators in the query collaboration group can improve the performance. As the threshold continues dropping below 0.45, the number of query collaborators reduces and the performance significantly drops until it arrives at the baseline performance of *tfidf* ranker (68.3%). It clearly shows that maintaining a controllable number of query collaborators can improve the performance. For the agglomerative clustering, it is the other story. As it continues merging from singleton clusters (corresponding to threshold 0) to one single cluster, the performance continues increasing until in the end it arrives the highest accuracy of 72.6%. However, unlike graph clustering, a peak never appears in the middle which may imply that agglomerative clustering is inferior to graph clustering.

The *max* function (Figure 4.2 (b)) leverages the strengths from the strongest collaborator in the group, which may potentially improve KB accuracy, but meanwhile hurt NIL accuracy.

As shown in the figure, as more collaborators join in the group, the performance increases first for both graph and agglomerative clustering, however, it starts to deteriorate when arriving at a threshold, and in the end, the performance drops even lower than the baseline of tfidf ranker.

The *min* function (Figure 4.2 (c)) leverages the strengths from the weakest collaborator in the group, which may potentially improve NIL accuracy, but meanwhile hurt KB accuracy. Our data analysis shows that the gain in NIL accuracy can not afford the larger loss in non-NIL accuracy, therefore, the performance continues dropping as the threshold increases. Min function is a counter example showing that searching query collaborators may not always lead to benefits.

To summarize so far, the best strategy for *tfidf* ranker in MiCR approach is *graph-ave* (applying graph clustering and using average function) which obtains overall accuracy gain of 5.3% over the baseline (68.3%). We further validate the performance of *graph-ave* using f_2, f_4 ranking functions, for *entity* ranker, we obtain accuracy gain of 6.3%, and for *profile* ranker, we obtain accuracy gain of 3.0%.

We then experiment the three supervised g_1 functions (ListNet, Maxent, and SVM respectively) using graph clustering as the query collaborator searching strategy. Figure 4.5 shows that ListNet, Maxent, SVM rankers obtain accuracy gain of 1.4%, 4.6%, 4.2% respectively over the baselines (corresponding to those points at threshold 0).

4.8.4 Impact of MaCR

Figure 4.3 shows that the MaCR approach obtains absolute accuracy gain of 1.3% (*voting* function) and 0.5% (*average* function) over the best baseline ranker (81.1%) when we add the 7th ranker (*entity* ranker). The improvement of *voting* function is statistically significant at a 99.6% confidence level by conducting Wilcoxon Matched-Pairs Signed-Ranks Test on the 10 folds of the testing set. However, the improvement of average function is not significant at the 0.05 level which implies that *average* is inferior to *voting*. We observe that the performance drops when there are even number of rankers in the ranker set using voting function, which implies that our tie breaking strategy may not be very effective.

We also experimented the *voting* function on the top 10 KBP2009 entity linking systems

(each system performance is shown in the table embedded in Figure 4.4, and experiment is similarly done as described in section 4.6). Figure 4.4 shows that it can obtain absolute accuracy gain of 4.7% over the top entity linking system (82.2%). The reasons why we achieve relative smaller gains using our own ranker set are as follows: (1) we use the same candidate object set for all rankers, while different KBP2009 systems may use their own set of objects. (2) our top 4 supervised rankers apply almost the same set of features, while different KBP2009 systems may apply more diversified features. Therefore, diversity is a highly important factor that makes MaCR approach effective.

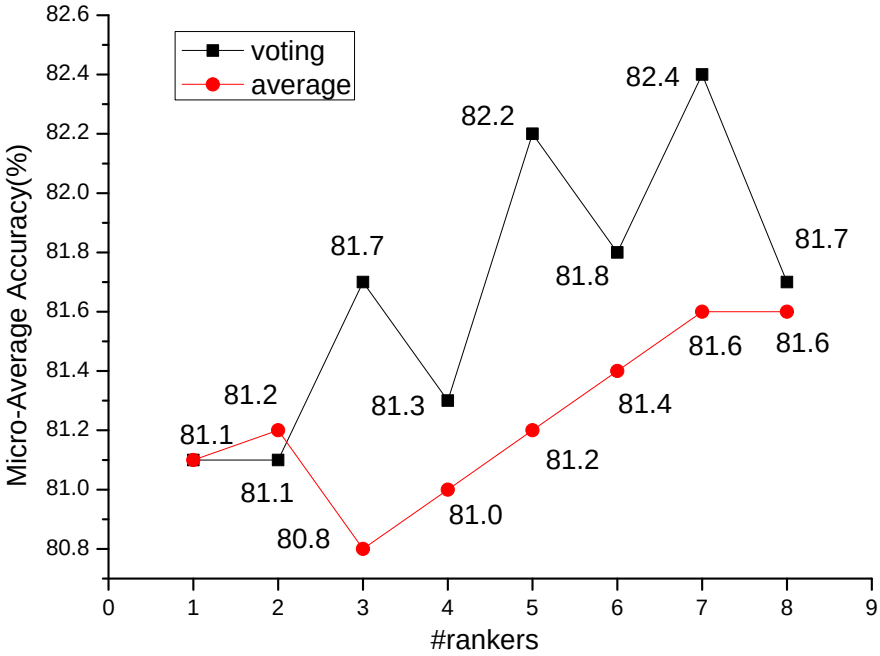


Figure 4.3: MaCR: comparison of voting and average.

4.8.5 Impact of MiMaCR

We applied the following settings in our MiMaCR approach: selecting graph clustering as the query collaborator searching strategy, including five rankers (tfidf, entity, Maxent, SVM and ListNet) in the ranker set, using *average* function to compute micro-ranking scores for the tfidf and entity ranker, using the three corresponding supervised versions of g_1 to compute micro-ranking scores for Maxent, SVM and ListNet respectively, and finally applying *voting* function to compute the macro-ranking score. In Figure 4.5, the curve of "MiMaCR" shows how the performance of MiMaCR is affected by the threshold in graph clustering. We obtain the best micro-average accuracy of 83.7% at threshold 0.3, which is 2.6 % higher than the best baseline

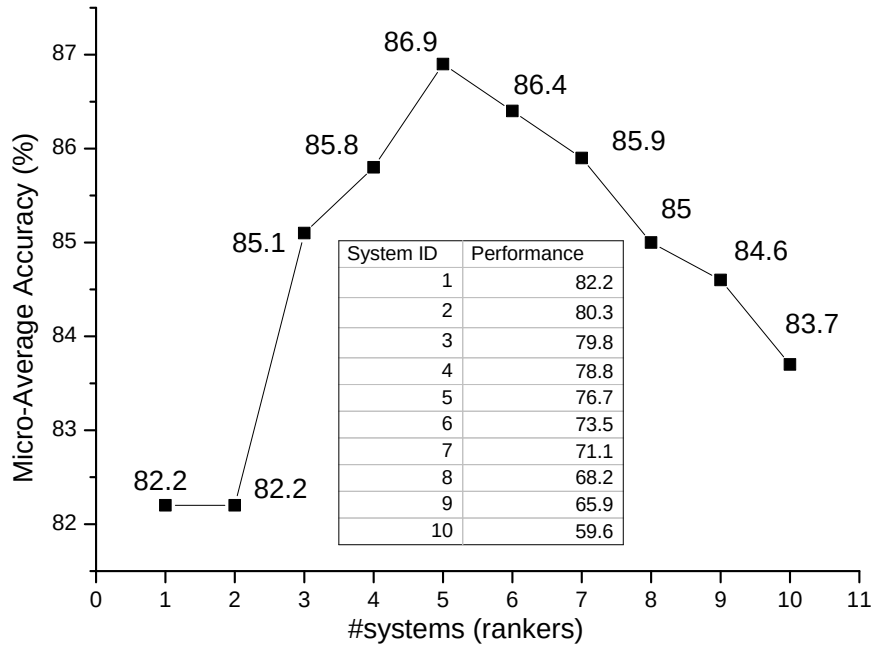


Figure 4.4: MaCR: voting function applied to the top 10 KBP2009 entity linking systems.

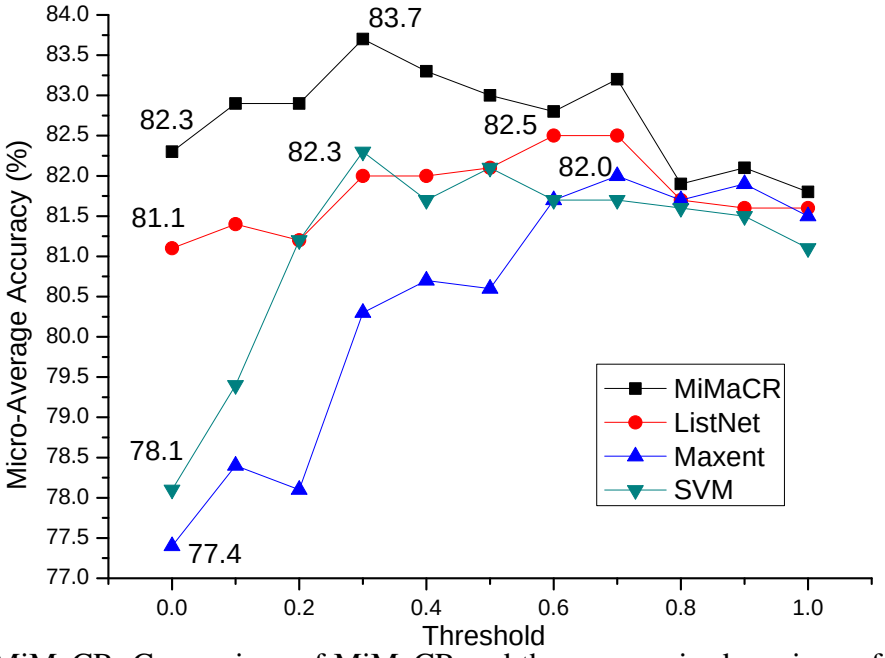


Figure 4.5: MiMaCR: Comparison of MiMaCR and three supervised versions of g_1 (ListNet, Maxent, and SVM respectively).

ranker (81.1%). The improvement is statistically significant at a 98.6% confidence level by conducting Wilcoxon Matched-Pairs Signed-Ranks Test on the 10 folds of the testing set. The score reported here is the second best in the KBP2010 evaluation.

4.9 Summary

In MiCR, effective searching of query collaborators and active interplay among members in the query collaboration group are two key factors that make MiCR successful. In MaCR, diversity is a highly important factor to make it successful. Overall, MiMaCR can bootstrap the performance to its maximum if integrating MiCR and MaCR properly. However, the better performance is at the expense of more computations.

Chapter 5

Collaborative Clustering

In this chapter, we propose a scheme that leverages instance collaborators and clusterer collaborators in order to obtain a better clustering.

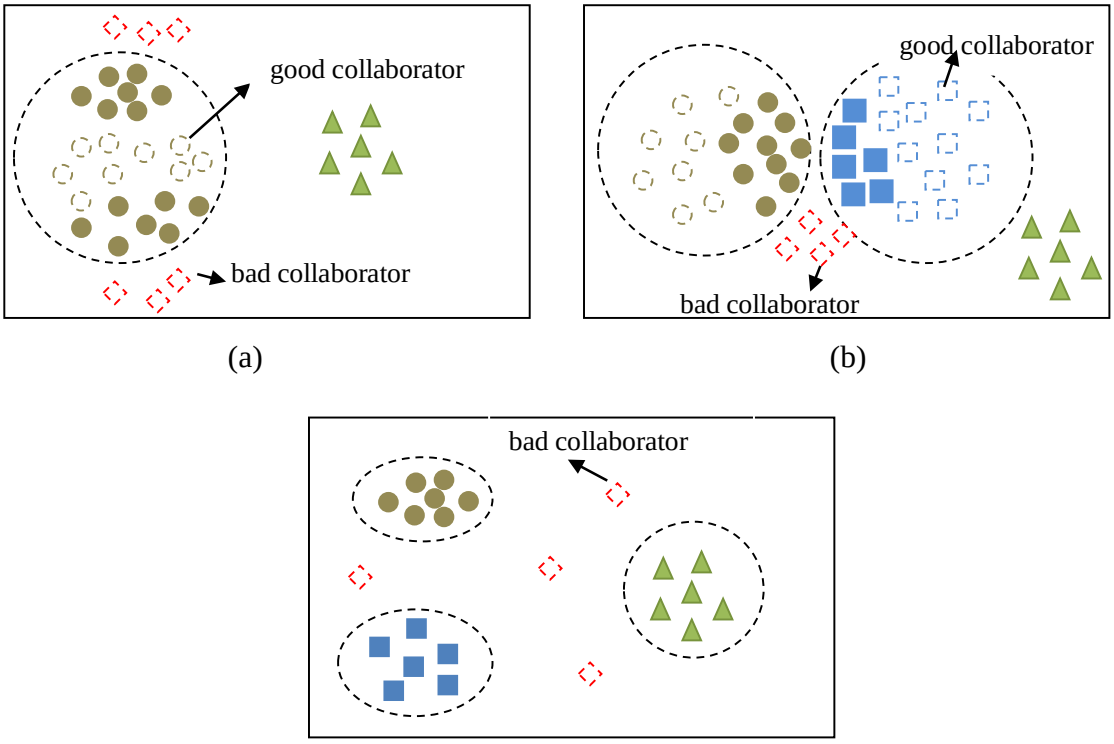


Figure 5.1: Examples showing good and bad collaborative instances

5.1 Motivations

5.1.1 Why Collaborative Instances Can Help?

The clustering problem is known as "unsupervised classification", because there is not a "supervisor" (as compared to the classification problem) that can tell which distribution an observation (instance) is from. For most clustering problems, we only focus on the instances that are required to be clustered and detect whether we can find good clustering structures from those instances. In this thesis, we assume that instances may come from a larger data space and we can populate other instances through some mechanism.

There are several scenarios in which we may not find good clustering structures based on the original set of instances:

(1) some instances look so dissimilar with some others, and at the first sight, they should come from two more more clusters, but they actually belong to a common good cluster, in other words, there is a hidden good cluster structure behind those instances, but unfortunately they are not representative enough for any clustering algorithm to discover. For example, in Figure 5.1 (a), instances are dispersed at the two ends of a circular shape cluster, it is likely that some clustering algorithm splits them into two clusters. However, after filling enough instances in the middle of the circular shape cluster, it is rather easy to find the natural cluster. A proverb describing this phenomenon is "seeing the trees instead of seeing the forests", so it may be easier to discover good clustering structures after looking at forests rather than some isolated trees.

(2) some instances look so similar with each other, and at the first sight, they should be clustered together, however, they actually come from two or more well formed clusters. For example, in Figure 5.1 (b), instances are dispersed at the ends of two circular shape clusters, and they are close enough, therefore, it is very likely that some clustering algorithm clusters all these instances together. However, after filling enough instances in the middle of the two circular shape clusters, it is rather easy to distinguish the two clusters. Another proverb describing this phenomenon is "seeing is deceiving", so in order not to be fooled by what they look, we should look at something more.

We have discussed two cases in which collaborative instances can help, but another question arises, "can adding collaborative instances always help?"

There are several cases that adding collaborative instances can not help, and sometimes, even hurt.

(1) Instances are representative enough and a clustering algorithm can detect good clustering structures only using those instances. For example, in Figure 5.1 (a), if collaborative instances are added among the detectable clusters, they can enhance the cluster structure, but probably may not improve the result. However, if collaborative instances (shown as bad collaborators in Figure 5.1 (a)) fall outside those well formed clusters, they look more like noises. In this case, a clustering algorithm can be disturbed so the result can become worse.

(2) By looking into the first case where collaborative instances can help, we can easily figure out a case in which they do not help, e.g., if the newly added instances fall outside the good cluster, it becomes even harder for a clustering algorithm to detect the good cluster (shown as bad collaborators in Figure 5.1 (b)).

(3) By looking into the second case where collaborative instances can help, we can also find a case when they do not help, e.g., if the newly added instances are similar with the confusable instances that are from two distinguished clusters, a clustering algorithm can be more confused (shown as bad collaborators in Figure 5.1 (c)).

Two important questions arise:

(1) how to select collaborative instances? A more specific question is what objective function needs to be optimized by continually adding more collaborative instances.We will answer this question in section 5.2.

(2) what is the algorithm for instance level collaborative learning? We will answer this question in section 5.3.

5.1.2 Why Collaborative Clusterers Can Help?

There is not a single clustering solution that works for every specific clustering problem and dataset.

(1) Every clustering algorithm has its own strengths and weaknesses. For example, K-

means algorithm is good at clustering spherical shapes of clusters, but can be very sensitive to noises and can not handle well with non-convex shapes of clusters. Most clustering algorithms aims at optimizing an objective function (which measures the clustering quality based on the information embedded in the data), however, it turns out that the optimization is a NP-Hard problem in most cases, and a solution using greedy optimization leads to a sub-optimal clustering.

(2) Distance function also plays an important role, however, it is also hard to obtain a consistently good distance function for every specific problem.

Therefore, we leverage the mass strengths from multiple clustering solutions in which we do not care much about which clustering solution is the best, and which is the worst. According to observations from common phenomena happened in our society (such as "voting" for a president), we have enough confidence to foretell that if appropriate combination scheme is applied, we can obtain a rather robust and better result in general.

5.2 Clustering Validation

The basic hypothesis of adding collaborative instances is that they can help uncover good clustering structures. Therefore, an important question is how to measure the quality of clustering structures such that good clustering can be distinguished from bad clustering and the best set of collaborative instances can be identified when the clustering quality is optimized.

Traditionally, according to whether it needs to rely on external information, the clustering validation measures can be mainly categorized into two types:

(1) **Internal** which measures the quality of a clustering structure by only using the information present in the data set.

(2) **External** which measures the quality of a clustering structure by comparing with external gold clustering.

We focus on internal validation measures in this section because we only use it for final evaluation.

5.2.1 Two Basic Criteria for a Good Internal Measure

There are two basic criteria that most of internal measures aim to satisfy [87].

1. **Cohesion**:

This criterion indicates how cohesive the instances in a cluster are. The more cohesive, the better the cluster is. If we use graph structure in which nodes represent instances and edge weights represent the distances (computed by a distance function d) of pairwise instances, then there are several ways to compute cohesion.

(1) The cohesion of a cluster C_i is computed as the sum of the weights of edges in the graph which connect points in the cluster, i.e.,

$$Cohesion(C_i) = \sum_{x \in C_i, y \in C_i} d(x, y) \tag{5.1}$$

The smaller value of this formula, the more cohesive the cluster is.

(2) If we can compute the centroid of a cluster, the cohesion of C_i can be computed as the sum of distances between each instance and the centroid, i.e.,

$$Cohesion(C_i) = \sum_{x \in C_i} d(x, c_i) \tag{5.2}$$

where c_i is the centroid of cluster C_i.

The smaller value of this formula, the more cohesive the cluster is.

Some researchers have used the term "compactness" to mean "cohesion" discussed here.

2. **Separation**:

This criterion indicates how separated a cluster is from other clusters.

(1)Using graph structure, separation between two clusters C_i and C_j can be computed as the sum of the weights of edges that connect the points from cluster C_i to the points from cluster C_j.

$$Separation(C_i, C_j) = \sum_{x \in C_i, y \in C_j} d(x, y) \tag{5.3}$$

The larger value of this formula, the more distinct or separated the two clusters are.

The separation of a cluster C_i can be computed as the sum of separations between cluster C_i to any other clusters

$$Separation(C_i) = \sum_{j \in \{1,...,k\}, j \neq i} Separation(C_i, C_j) \tag{5.4}$$

(2)Separation between two clusters can also be computed as the the distance of the two centroids in the two clusters, i.e.,

$$Separation(C_i, C_j) = d(c_i, c_j) \tag{5.5}$$

where c_i and c_j are the centroids of C_i and C_j respectively.

The larger value of this formula, the more distinct the two clusters are.

5.2.2 Overview of Internal Validation Measures

Most of internal validation measures proposed so far have captured either one or both of the criteria. However, it is a consensus that a good internal validation measure should capture both. In this thesis, we intend to study the following 12 representative internal validation measures that have been widely used in previous work [41, 61, 64, 90, 100].

First, We define the common notations in Table 5.1.

Notation	Meaning
D	data set
d	number of dimensions for each instance in D
n	the number of instances in the data set
k	the number of clusters
c	centroid of the whole data set
C_i	the i^{th} cluster
c_i	centroid of cluster C_i
n_i	the number of instances in cluster C_i
x_j	the j^{th} instance in a cluster
$d(x, y)$	distance between instance x and y
$\|X\| = \sqrt{XX^T}$	dot product, where X is a row vector

Table 5.1: Notations of internal validation measures

We first summarize the 12 internal validation measures in Table 5.2, and then discuss the details of each measure. For convenience of discussions, we use distance functions in the formula, however, they can be replaced with similarity functions when needed. It is worth

noting that in most cases, minimizing some internal measure using distance function means maximizing the measure using similarity function, and vice versa as shown in the column of "Optimal(distance)" and "Optimal(similarity)". There is only one exception, when Silhouette Coefficient is applied, we need to maximize it either using distance function or similarity function.

Name(Notation)	Optimal(distance)	Optimal(similarity)	Complexity
I_1	Min	Max	$O(n^2)$
I_2	Min	Max	$O(n)$
ε_1	Max	Min	$O(k)$
H_1	Min	Max	$O(n^2)$
H_2	Min	Max	$O(n)$
G_1	Max	Min	$O(n^2)$
Calinski-Harabasz index (CH)	Max	Min	$O(n)$
Dunn's Index (D)	Max	Min	$O(n^2)$
Silhouette Coefficient (SC)	Max	Max	$O(n^2)$
Davies-Bouldin Index (DB)	Min	Max	$O(n)$
SD validity index (SD)	Min	Max	$O(n)$
S_Dbw Validity Index (S_Dbw)	Min	Max	$O(n^2)$

Table 5.2: Summary of 12 internal validation measures

1. I_1.

This measure is computed as

$$I_1 = \sum_{i=1}^{k} n_i \left(\frac{1}{n_i^2} \sum_{x,y \in C_i} d(x,y) \right) \tag{5.6}$$

I_1 computes the sum of average pairwise distances between two instances in the same cluster, therefore, it measures the degree of cohesion. The smaller the value is, the more cohesive the clusters are. Therefore, the optimal clustering is obtained when this measure is minimized.

2. I_2.

This measure is computed as

$$I_2 = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, c_i) \tag{5.7}$$

I_2 computes the sum of pairwise distances between an instance and the centroid in the same cluster, therefore, it also measures the degree of cohesion. The smaller the value is, the more cohesive the clusters are. Therefore, the optimal clustering is obtained when this measure is minimized.

This measure is actually the basic objective function which k-means clustering aims to optimize.

3. ε_1. This measure is computed as

$$\varepsilon_1 = \sum_{i=1}^{k} n_i d(c_i, c) \tag{5.8}$$

ε_1 computes the sum of distances between a cluster centroid and the centroid of the whole data set, therefore, it measures the separation of clusters. The larger the value is, the more separated the clusters are. Therefore, the optimal clustering is obtained when this measure is maximized.

4. H_1. This measure is computed as

$$H_1 = \frac{I_1}{\varepsilon_1} \tag{5.9}$$

H_1 combines I_1 and ε_1 which captures both cohesion and separation. Therefore, the optimal clustering is obtained when this measure is minimized.

5. H_2. This measure is computed as

$$H_2 = \frac{I_2}{\varepsilon_1} \tag{5.10}$$

H_2 combines I_2 and ε_1 which captures both cohesion and separation. The optimal clustering is obtained when this measure is minimized.

6. G_1. This measure is computed as

$$G_1 = \sum_{i=1}^{k} \frac{cut(C_i, X \backslash C_i)}{\sum\limits_{x,y \in C_i} d(x,y)} = \sum_{i=1}^{k} \frac{\sum\limits_{x \in C_i, y \notin C_i} d(x,y)}{\sum\limits_{x,y \in C_i} d(x,y)} \tag{5.11}$$

In this formula, $cut(C_i, X \backslash C_i)$ computes the sum of distances between instances in cluster C_i and instances outside cluster C_i, so it measures separation . Therefore, the optimal clustering is obtained when this measure is maximized.

This metric was proposed as an objective function for spectral clustering based on normalized cut [82].

7. ***Calinski-Harabasz index (CH)*** [7].

This measure is computed as follows:

$$CH = \frac{T_B}{T_W} \times \frac{n-k}{k-1} \tag{5.12}$$

where

$$T_B = \sum_{i=1}^{k} n_i d^2(c_i, c) \tag{5.13}$$

$$T_W = \sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, c_i) \tag{5.14}$$

In fact, T_B indicates the inter-cluster variances, hence it measures the degree of separation. The larger the value is, the more separated the clusters are. T_W indicates intra-cluster variances, hence it measures the degree of cohesion. The smaller the value is, the more cohesive the clusters are. A cohesive and separated clustering should have large T_B and small T_W.

CH looks very similar with H_2 except that it multiplies a normalization term $\frac{n-k}{k-1}$. It is worth noting that this measure will be undefined when $k = 1$ or $k = n$ because of $0/0$.

8. ***Dunn's Index (D)*** [32]

This measure is computed as:

$$DN = \frac{\min_i \min_{j,j \neq i} \min_{x \in C_i, y \in C_j} d(x, y)}{\max_m \max_{x \in C_m, y \in C_m} d(x, y)} \tag{5.15}$$

The numerator in the formula indicates inter-cluster separation which is computed as the overall minimum pairwise distance between an instance in one cluster and an instance in the other cluster. The larger the value is, the more distinct the clusters are. The denominator indicates intra-cluster cohesion which is computed as the overall maximum pairwise distance between two instances in one cluster[1]. The smaller the value is, the more cohesive the clusters are. Overall, a large value of Dunn's index means a cohesive and distinct clustering structure. Therefore, the optimal clustering is obtained when the Dunn's Index is maximized. It is worth noting that this measure will be undefined when $k = 1$ (numerator can not be computed) or $k = n$ (the denominator is 0).

9. *Silhouette Coefficient (SC)* [78]

The Silhouette Coefficient of the clustering is computed as the average of Silhouette Coefficient for each cluster, i.e.,

$$SC = \frac{1}{k} \sum_{i=1}^{k} SC_i \tag{5.16}$$

where SC_i is the Silhouette Coefficient for cluster C_i.

[1] it is also called the diameter of a cluster

The value of SC_i is then computed as the average of Silhouette Coefficient for each instance in the cluster, i.e.,

$$SC_i = \frac{1}{n_i} \sum_{j=1}^{n_i} SC_j \tag{5.17}$$

where n_i is the number of instances in cluster C_i, and SC_j is the Silhouette Coefficient for instance x_j in cluster C_i.

To compute Silhouette Coefficient for each instance x_j,

(1) For an instance x_j in cluster C_i, compute the average distance (a_j) to the other instances in the same cluster, i.e.,

$$a_j = \frac{1}{n_i - 1} \sum_{y \in C_i, y \neq x_j} d(x_j, y) \tag{5.18}$$

(2) For the instance x_j in cluster C_i, compute the minimum of the average distance (b_j) to the instances in another cluster C_m ($m \neq i$), i.e.,

$$b_j = \min_{m \in \{1,\ldots,k\}, m \neq i} \{\frac{1}{n_m} \sum_{y \in C_m} d(x_j, y)\} \tag{5.19}$$

(3) The Silhouette Coefficient for instance x_j is then computed as

$$SC_j = (b_j - a_j) / \max(b_j, a_j) \tag{5.20}$$

The Silhouette Coefficient of an instance SC_j, a cluster SC_i, and a clustering SC varies from -1 to 1. When SC_j is closer to 1, it means that instance x_j is an instance that contributes a lot to the cohesion and separation of the cluster it belongs to. When SC_i is closer to 1, it means that cluster C_i is a cohesive cluster and it is distinct to any other clusters. When SC is closer to 1, it means that all the clusters are cohesive and distinct from each other.

Therefore, the optimal clustering is obtained when the Silhouette Coefficient SC is maximized. It is worth noting that this measure will be undefined when $k = 1$ (b_j can not be computed) or $k = n$ (a_j can not be computed).

10. ***Davies-Bouldin Index (DB)*** [27]

This measure is computed as

$$DB = \frac{1}{k}\sum_{i=1}^{k} D_i \tag{5.21}$$

where

$$D_i = \max_{j,j\neq i} D_{i,j} \tag{5.22}$$

and

$$D_{i,j} = [\frac{1}{n_i}\sum_{x\in C_i} d(x, c_i) + \frac{1}{n_j}\sum_{x\in C_j} d(x, c_j)]/d(c_i, c_j) \tag{5.23}$$

Term $D_{i,j}$ computes the intra-cluster cohesion and inter-cluster separation involving cluster C_i and C_j. Hence, the smaller value of $D_{i,j}$, the more cohesive and distinct C_i and C_j are. Term D_i represents the worse case involving cluster C_i which is computed as the maximum value of $D_{i,j}$. The value of Davies-Bouldin index is then computed as the average of D_i for all the clusters. A small value of this index means a cohesive and distinct clustering structure. Therefore, the optimal clustering is obtained when the Davies-Bouldin Index is minimized (but should be larger than 0). It is worth noting that when $k = 1$, the value of this index is undefined and when $k = n$, the value is 0.

11. ***SD validity index (SD)*** [40]

This measure is computed as:

$$SD = \alpha \cdot Scatt + Dis \tag{5.24}$$

where

$$Scatt = \frac{1}{k}\sum_{i=1}^{k} \frac{\|\sigma(C_i)\|}{\|\sigma(D)\|} \tag{5.25}$$

$$Dis = \frac{\max\limits_{i,j\neq i} d(c_i, c_j)}{\min\limits_{i,j\neq i} d(c_i, c_j)} \sum_{i=1}^{k} \frac{1}{\sum\limits_{j=1,j\neq i}^{k} d(c_i, c_j)} \tag{5.26}$$

and

α is a weighting factor which is equal to $Dis(n_{max})$ in which n_{max} is the maximum number of clusters.

The first term $Scatt$ computes the average scattering (cohesion) of clusters and the smaller the value is, the more cohesive the clusters are. The second term Dis computes the total separation of clusters and the smaller the value is, the more distinct the clusters are. Overall, a small value of SD validity index indicates a cohesive and distinct clustering. Therefore, the optimal clustering is obtained when this index is minimized. It is worth noting that [40] has shown that the value of n_max does not have significant influence on the value of SD validity index.

In the first term, $\sigma(D)$ is defined as the variance of the data set, $\sigma(C_i)$ is the variance of the cluster C_i, and $\|\cdot\|$ is dot product.

σ_D is a column vector

$$\sigma_D = \begin{bmatrix} \sigma_D^1 \\ ... \\ \sigma_D^d \end{bmatrix} \tag{5.27}$$

and the p^{th} dimension in the column vector

$$\sigma_D^p = \frac{1}{n}\sum_{i=1}^{n}(x^p - c^p)^2 \tag{5.28}$$

σ_{C_i} is a column vector

$$\sigma_{C_i} = \begin{bmatrix} \sigma_{C_i}^1 \\ ... \\ \sigma_{C_i}^d \end{bmatrix} \tag{5.29}$$

and the p^{th} dimension in the column vector

$$\sigma_{C_i}^p = \frac{1}{n_i}\sum_{j=1}^{n_i}(x^p - c_i^p)^2 \tag{5.30}$$

12. ***S_Dbw Validity Index (S_Dbw)*** [39]

This measure is computed similarly as SD validity index, but it differs in the computation of inter-cluster separation by taking into density. The optimal clustering is obtained when this index is minimized.

$$S\_Dbw = Scatt + Dens\_bw \tag{5.31}$$

$Scatt$ is computed exactly as the one in SD validity index, and $Dens\_bw$ is computed as

$$Dens\_bw = \frac{1}{k(k-1)} \sum_{i=1}^{k} \sum_{j=1, j\neq i}^{k} \frac{density(u_{ij})}{\max\{density(c_i), density(c_j)\}} \tag{5.32}$$

where c_i, c_j are the centers of clusters C_i and C_j respectively, and u_{ij} is the center point of the line connecting c_i and c_j. The term $density$ is defined as

$$density(u_{ij}) = \sum_{l=1}^{n_{ij}} f(x_l, u_{ij}) \tag{5.33}$$

where x_l is defined as an instance in a set of S which consists of instances in both C_i and C_j, i.e., $x_l \in C_i \cup C_j = S$ and n_{ij} is the total number of points that belong to the clusters C_i and C_j.

The f function is defined as

$$f(x, u) = \begin{cases} 0 & \text{if } d(x, u) > std \\ 1 & \text{otherwise} \end{cases} \tag{5.34}$$

in which The standard deviation std is defined as

$$std = \frac{1}{k} \sqrt{\sum_{i=1}^{k} \|\sigma(C_i)\|} \tag{5.35}$$

and $\sigma(C_i)$ is defined in 5.29.

13. ***Other measures***

In the literature, researchers have proposed variants of the above mentioned measures, e.g., Simplified Silhouette Coefficient by replacing the distance between two instances with the distance between the instance and the centroid; and some other measures, e.g., Root-mean-square standard deviation [80], Modified Hubert Γ statistic [47], I index [64], Xie-Beni index [93]. In

previous work, clustering problem is modeled using graph structure. Besides the $\mathcal{G}_1$ discussed earlier, there are quite some other measures proposed in the context of graph, as surveyed in our previous work [13].

All the internal measures proposed so far can be used to evaluate the quality of clustering without referring to external gold clustering. The question is that given all these internal measures, which one should we choose? Are there some internal measures that can outperform the others? We will focus on examining various internal measures in the next section.

5.2.3 Comparison of Various Internal Measures

Many researchers have started to compare various internal measures since last eighties [41, 61, 68]. Most of previous work focused on comparing those measures by identifying whether they can identify the correct number of clusters. An internal measure is normally considered good if it can correctly find the number of clusters, i.e., reaching its optimal value (either maximum or minimum) at the correct number of clusters.

A general procedure of determining the best number of clusters using an internal measure is as follows:

Algorithm 5 Determine the best number of clusters using an internal measure.

Input:
a clustering algorithm $\mathcal{F}$;
an internal validation measure $\mathcal{M}$;
a data set D;
n: number of instances in D;
n_{max}: maximum number of clusters, suggested value $n_{max} = \sqrt{n}$

Output:
1: **for** $k = 2 \rightarrow n_{max}$ **do**
2: Apply $\mathcal{F}$ on D using parameter k and produce a clustering $\mathcal{C}$
3: Compute the clustering quality m_k on $\mathcal{C}$ using $\mathcal{M}$
4: **end for**
5: Find the optimal value in the array of m_k and the optimal k_{opt} when m_k is optimal
6: **return** k_{opt}

Recently, Liu et al. [61] compared 11 measures from the following five aspects:

(1) monotonicity: whether a measure can reach a optimal value at some certain number of cluster k, rather than increase or decrease monotonically.

(2) noise: whether a measure can identify the correct number of clusters when there are noise instances dispersed among well separated clusters.

(3) density: whether a measure can identify the correct number of clusters when there are dense regions.

(4) subclusters: whether a measure can identify the correct number of clusters when there are clusters that are closed to each other.

(5) skewed Distributions:whether a measure can identify the correct number of clusters when the cluster size follows a skewed distribution, i.e., some clusters have much more instances than the others.

They found that only S_Dbw Validity Index can identify the correct number of clusters for all the five cases based on the synthetic datasets they developed.

Although it is indeed important to identify the correct number of clusters, the problem is whether it is the only aspect that can distinguish a good internal measure from a bad one. Vendramin et al. [90] have argued that there are some conceptual flaws in the comparison paradigm by only considering whether a good measure can identify the correct number of clusters. The key issue is that there are many clustering results that produce the correct number of clusters, but are actually not optimal and there are many clustering results that produce the wrong number of clusters, but actually exhibit good clustering structure. Therefore, they proposed a method that compares multiple measures by computing the correlation (e.g., Pearson Correlation) between the internal measure with some external measure. The hypothesis is that a good internal measure should have a high degree of compatibility with external measure. The problem is that if the external measure is a flawed one, high correlation may turn out to be a misleading result.

In this thesis, we will examine the 12 internal measures from another aspect: can they help recover the good clustering structure by incrementally adding more collaborative instances? In the next section 5.3, we will elaborate how to use an internal measure for instance level collaborative clustering (MICC).

5.3 Micro Collaborative Clustering (MiCC)

Micro collaborative clustering (i.e.,instance-level collaborative clustering) takes the advantage of looking into a bigger and better "picture" of the underlying structure by adding collaborative instances. In order to achieve success in MiCC, we need to find solutions for the following key problems.

5.3.1 Mechanism of Populating Collaborative Instances

In general, this is a difficult problem, since for clustering, we do not know the function of data distribution. However, when it comes to specific applications, it is possible. The solutions can vary from case to case. For example, in name entity clustering, each instance contains a name and a context document, in order to find collaborative instances, we retrieve a certain number of documents from a large corpus and each of the document contains the name, e.g., in order to cluster "Michael Jordan" in different documents, we just retrieve more documents containing "Michael Jordan". We define the set of collaborative instances as Y.

5.3.2 Selection of Collaborative Instances

It is possible that not all collaborative instances in Y are helpful, and some can even hurt. In order to identify the best set of collaborative instances, we apply an iterative algorithm in which for each iteration, we pick some collaborative instances from the pool and check whether they can improve the clustering structure of the original data set (using an internal measure). So the key issue is how to select collaborative instances.

In this thesis, we apply the following simple strategy: *in each iteration*, repeat a procedure of randomly selecting a certain number of collaborative instances for some rounds, and pick one of the best set of randomly selected collaborative instances which produced the best clustering measured by an internal measure.

5.3.3 Algorithms of MiCC

The basic idea of MiCC is that we incrementally add instances from the pool of Y, continually check the clustering quality of the original set of instances after new instances are added, and identify the optimal clustering quality at some iteration. In this thesis, we apply a simple strategy to select collaborative instances by random selection (Algorithm 6).

Algorithm 6 MiCC algorithm by random selection.

Input:

$X = \{x_1, x_2, ..., x_n\}$: instances in the data set;
$Y = \{y_1, y_2, ..., y_m\}$: a pool of candidate collaborative instances;
$\mathcal{F}$: a clustering algorithm;
$\mathcal{M}$: an internal measure; //Assume the measure takes maximum as optimal
n_{step}: maximum number of collaborative instances picked in each iteration;
n_{trials}: number of times
$n_{iterations}$: number of iterations

Output:

a set of best collaborative instances: $BestC$;
the optimal value computed by the internal measure LM_{opt};

1: Apply $\mathcal{F}$ on X and output a clustering, compute the clustering quality by the internal measure $\mathcal{M}$.
2: initialize a list of best found candidate collaborative clustering instances LC
3: initialize a list of best found values of clustering quality LM
4: **for** $i = 1 \rightarrow n_{iterations}$ **do**
5: **for** $j = 1 \rightarrow n_{trials}$ **do**
6: Randomly pick n_{step} collaborative clustering instances (naming the set as Y_j) from Y and produce a new set of instances by $X \cup Y_j$.
7: Apply $\mathcal{F}$ on $X \cup Y_j$, compute clustering quality LM_ij using $\mathcal{M}$.
8: **end for**
9: Find LM_ik such that $LM_{ik} >= LM_{ij}$ for $j \in \{1, ..., n_{trials}\}$, expand X such that $X = X \cup Y_k$, remove Y_k from Y such that $Y = Y - Y_k$.
10: $LC_i = Y_k$, $LM_i = LM_{ik}$
11: **end for**
12: Find LM_u such that $LM_u >= LM_i$ for $i \in \{1, ..., n_{iterations}\}$
13: $LM_{opt} = LM_u$
14: $BestC = LC_1 \cup ... \cup LC_u$
15: **return** $BestC$ and LM_{opt}

5.3.4 Obtaining Final Clustering Solution

The original set of instances is a subset of expanded set including the best selected collaborative instances, therefore, the clustering result can be retrieved from the clustering based on the

expanded set, in other words, we only care about the cluster ids that are assigned to the original set of instances. The evaluation (using external measure) is still conducted on the original instances.

5.4 Macro Collaborative Clustering (MaCC)

Macro collaborative clustering (i.e., clusterer-level collaborative clustering) takes the advantage of leveraging the integration of various clustering results from different clustering methods by applying a consensus function. There are two key issues in MaCC:

1. Ensemble generation: A clustering ensemble is a set of clusterings, each of which is generated by a clusterer. Diversified clusterers can be implemented through:

(1) different clustering algorithms;

(2) different distance functions to compute the distance between two instances;

(3) different parameter settings for a specific algorithm;

(4) different dimension reduction methods which project from high dimensional space to lower dimensional space;

(5) different sampling methods;

We denote the clustering ensemble as $\Pi = \{\pi^1, ..., \pi^r\}$ in which r is the number of clusterers and each clustering π^i consists of k_i number of clusters, i.e., $\pi^i = \{C_1^i, ..., C_{k_i}^i\}$ where $C_1^i \cup ... \cup C_{k_i}^i = X$.

In section 5.4.1, we will discuss various clusterers we used.

2. Consensus function: Given the clustering ensemble $\Pi = \{\pi^1, ..., \pi^r\}$, a consensus function Γ maps the ensemble to an integrated clustering, i.e., $\Gamma : \Pi \rightarrow \mathcal{C}$. In section 5.4.2, we will discuss some major categories of consensus functions.

5.4.1 Ensemble Generation

In this thesis, we apply the following series of clustering algorithms

5.4.1.1 K-means algorithm

The algorithm (Algorithm 7) works as follows:

(1) Randomly select k instances as centroids.

(2) Assign each instance to a cluster label associated with its closest centroid.

(3) Recompute centroids.

(4) repeat (2) and (3) until the centroids do not change any more.

Algorithm 7 K-means algorithm.

Input:
$X = \{x_1, x_2, ..., x_n\}$: instances in the data set
k: number of clusters
$d(x, y)$: a distance function which computes distance between vector x and y
$rand(X, k)$: a function that randomly picks k instances from X

Output:
$\{c_1, ..., c_k\}$: set of k centroids $L = \{l(x_1), ..., l(x_n)\}$: set of cluster labels of X

1: $\{c_1, ..., c_k\} = rand(X, k)$.
2: **for** $i = 1 \rightarrow n$ **do**
3: $l(\mathrm{x}_i) = \underset{j,j\in\{1,...,k\}}{\arg\min}\ \mathrm{d}(\mathrm{x}_i,\mathrm{c}_j)$
4: **end for**
5: changed = true
6: **while** changed **do**
7: **for** $i = 1 \rightarrow k$ **do**
8: $S = \{j | j \in \{1, ..., n\}, l(x_j) = i\}$
9: $c_i = \frac{1}{|S|} \sum_{j \in S} x_j$
10: **end for**
11: changed = false
//Assign each instance to its closest centroid
12: **for** $i = 1 \rightarrow n$ **do**
13: $label_{new} = \underset{j,j\in\{1,...,k\}}{\arg\min}\ \mathrm{d}(\mathrm{x}_i,\mathrm{c}_j)$
14: **if** $label_{new} \neq l(x_i)$ **then**
15: $l(x_i) = label_{new}$
16: changed = true
17: **end if**
18: **end for**
19: **end while**
20: **return** $L = \{l(x_1), ..., l(x_n)\}$

We can obtain multiple K-means based clusterings by varying the selection of initial centroids, and different choices of distance functions.

K-means tends to run fast with the running complexity of $O(knt)$ where k is the number of

clusters, n is the number of instances, and t is the number of iterations (normally, $k \ll n$ and $t \ll n$). Meanwhile, there are several concerns related with K-means, (1) it tends to work well for spherical shaped clusters of similar sizes, but does not work well for non-convex shapes or clusters of different sizes; (2) it is quite sensitive to noisy instances, because an outlier is forced to be in a cluster even if it is far from the centroid , thus distort the cluster shape; (3) it is sensitive to the initial centroid selection, thus tends to get trapped in local optima values.

5.4.1.2 Agglomerative clustering algorithms[84]

Two versions of agglomerative clustering algorithms are applied here, one is based on linkage of clusters (Algorithm 8), the other is based on optimizing a criterion function (Algorithm 9).

Agglomerative clustering by linkage.

(1) Compute the distance (guided by some linkage criterion) for each pair of clusters (initially each cluster is an instance).

(2) Merge two clusters with the smallest distance.

(3) Repeat (1)(2) until the number of clusters k is reached.

When computing the distance of two clusters, normally, we have three strategies: (a) single-linkage which computes the minimum pairwise distance in the two clusters; (b) complete-linkage which computes the maximum pairwise distance in the two clusters; (c) average-linkage which computes the average pairwise distance in the two clusters.

Agglomerative clustering by optimizing an internal metric.

(1) Merge two clusters which leads to the highest gains computed by an internal metric.

(2) Repeat (1) until the number of clusters k is reached.

There are two major concerns related with agglomerative clustering, (1) the running complexity is relatively high, which takes $O(n^3)$ in general but can be reduced to $o(n^2)$ for single-linkage clustering and complete-linkage clustering; (2) errors in early stages of merging can not be fixed in the later iterations which can be a serious problem if there are lots of close but actually distinct clusters.

Algorithm 8 Agglomerative clustering algorithm (by linkage).

Input:
$X = \{x_1, x_2, ..., x_n\}$: instances in the data set
k: number of clusters
$d(C_i, C_j)$: a distance function which computes distance between cluster C_i and C_j
Output:
$\mathcal{C} = \{C_1, ..., C_k\}$: set of clusters

$\mathcal{C} = \emptyset$
for $i = 1 \rightarrow n$ **do**
  $C_i = \{x_i\}$
  $\mathcal{C} = \mathcal{C} \cup \{C_i\}$
end for
while $|\mathcal{C}| > k$ **do**
  $(C_u, C_v) = \arg\min_{C_i, C_j \in \mathcal{C}} d(C_i, C_j)$
  $\mathcal{C} = (\mathcal{C} \setminus \{C_u\} \setminus \{C_v\}) \cup \{C_u \cup C_v\}$
end while
return $\mathcal{C}$

Algorithm 9 Partitioning clustering algorithm (optimizing an internal metric).

Input:
$X = \{x_1, x_2, ..., x_n\}$: instances in the data set
k: number of clusters
$f(\mathcal{C})$: an internal metric which computes a clustering $\mathcal{C}$, assume optimal when minimizing
Output:
$\mathcal{C} = \{C_1, ..., C_k\}$: set of clusters

$\mathcal{C} = \emptyset$
for $i = 1 \rightarrow n$ **do**
  $C_i = \{x_i\}$
  $\mathcal{C} = \mathcal{C} \cup \{C_i\}$
end for
while $|\mathcal{C}| > k$ **do**
  $(C_u, C_v) = \arg\min_{C_i, C_j \in \mathcal{C}} f((C \setminus \{C_i\} \setminus \{C_j\}) \cup \{C_i \cup C_j\})$
  $\mathcal{C} = (\mathcal{C} \setminus \{C_u\} \setminus \{C_v\}) \cup \{C_u \cup C_v\}$
end while
return $\mathcal{C}$

5.4.1.3 Partitional clustering algorithms[84]

Two versions of partitional clustering algorithms are applied in this thesis, one is based on *repeated bisections*(Algorithm 10), the other is based on *direct k-way partitioning* (Algorithm 11).

In repeated bisections, the whole set of instances are continually bisected until k clusters are obtained. Each bisection iteration consists of two major steps: (a) initial clustering (b) cluster

refinement. In initial clustering, two instances are selected as seeds of two clusters, and the other instances are assigned to one of the two clusters according to the distance between the instance and the seed. Then in the cluster refinement step, it consists of a number of iterations. During each iteration, the instances are visited in random order. For each instance, validate whether we can obtain improvement in the value of an internal metric by moving it to the other cluster. If improvement can be obtain, move the instance to the other cluster immediately, otherwise, just leave it in the old cluster. This refinement step stops when it reaches the number of iterations or in some iteration, no instances can be moved. An important issue related with bisection is to determine which cluster to be bisected next. A simple strategy is to always select the largest cluster available at that point of bisection. This strategy tends to output balanced cluster sizes and can not work well on data set in which cluster sizes are skewed. Another strategy is to select a cluster which can lead to better improvement in the value of an internal metric if it is bisected.

A direct k-way partitioning also consists of two major steps, which are quite similar with the two steps in a bisection iteration, but differs in (1) initial clustering: instead of choose 2 seeds, k seeds are chosen and instances are assigned to the k seeds accordingly; (2) cluster refinement: instead of moving an instance to the other cluster if there is improvement, moving the instance to one of the $k-1$ which leads to the best improvement in the value of an internal metric. It is worth noting that if the internal metric I_2 (optimizing the sum of pairwise distances between an instance and the centroid in the same cluster) is applied, it is exactly the general K-means algorithm.

The approaches applied in the step of cluster refinement are greedy in nature, and by no means lead to global optimal because the refinement is based on randomly selected seeds. In order to reduce the sensitivity, we can repeat the whole procedure for a number of trials, and select the best clustering which obtains the maximum value computed by the internal metric.

5.4.2 Consensus Functions

A consensus function maps a set of clusterings into a final clustering. Various approaches have been proposed, including mutual information based [62, 88], voting based [34, 35], co-

Algorithm 10 Agglomerative clustering algorithm (repeated bisection).

Input:

$X = \{x_1, x_2, ..., x_n\}$: instances in the data set

k: number of clusters

$f(\mathcal{C})$: an internal metric which computes the quality of clustering $\mathcal{C}$, assume optimal when minimizing

Output:

$\mathcal{C} = \{C_1, ..., C_k\}$: set of clusters

1: $\mathcal{C} = \{X\}$ //select the largest cluster to bisect
2: **while** $|\mathcal{C}| < k$ **do**
3: $B = \arg\max_{C_i \in \mathcal{C}} |C_i|$
4: $(C_u, C_v) = \arg\min_{C_i \cup C_j \in B, C_i \cap C_j = \emptyset} f((\mathcal{C} \setminus \{B\}) \cup \{C_i\} \cup \{C_j\})$
5: **end while**
6: **return** $\mathcal{C}$

Algorithm 11 Partitioning clustering algorithm (k-way direct).

Input:

$X = \{x_1, x_2, ..., x_n\}$: instances in the data set

k: number of clusters

$f(\mathcal{C})$: an internal metric which computes the quality of a clustering $\mathcal{C}$, assume optimal when minimizing

Output:

$\mathcal{C} = \{C_1, ..., C_k\}$: set of clusters

1: $\mathcal{C} = \{X\}$
2: $(C_1, C_2, ..., C_k) = \arg\min_{C_1 \cup ... \cup C_k = X, \forall i,j, C_i \cap C_j = \emptyset} f(\{C_1\} \cup ... \cup \{C_k\})$
3: **return** $C_1, C_2, ..., C_k$

association matrix based [36], mixture model based [89], and graph based [33, 85].

5.4.2.1 Consensus function based on mutual information

Without knowing the relative importance (confidence) of those multiple clusterings, a reasonable way is to find the best clustering which shares most mutual information with all the clusterings in the ensemble. Strehl et. al. [85] applied normalized mutual information metric and proposed the following consensus function which aims to optimize an average normalized mutual information (ANMI), i.e.,

$$\pi^{(opt)} = \arg\max_{\hat{\pi}} \frac{1}{r} \sum_{i=1}^{r} NMI(\hat{\pi}, \pi^i) \tag{5.36}$$

where NMI is defined as

$$NMI(\pi_i, \pi_j) = \frac{MI(\pi_i, \pi_j)}{\sqrt{H(\pi_i)H(\pi_j)}} \tag{5.37}$$

MI is mutual information between two clusterings

$$MI(\pi_i, \pi_j) = \sum_{u=1}^{k_i} \sum_{v=1}^{k_j} \frac{n_{uv}}{n} \log \frac{n \cdot n_{uv}}{n_u \cdot n_v} \tag{5.38}$$

where n is the number of instances in the data set, k_i and k_j are the number of clusters in clustering π^i and π^j respectively, n_u and n_v are the number of instances in cluster C_u^i (in clustering π^i) and C_v^j (in clustering π^j) respectively, n_{uv} is the number of common instances in cluster C_u^i and C_v^j.

H is entropy of a clustering

$$H(\pi^i) = \sum_{u=1}^{k_i} \frac{n_u}{n} \log \frac{n_u}{n} \tag{5.39}$$

In order to solve the optimization problem in equation 5.36, we can apply a brute-force search going over all possible clusterings ($\hat{\pi}$) and find out the maximum. However, the searching space is $k^n/k!$ for $k >> n$ which makes the problem intractable. [85] proposed a greedy search method which first picks a clustering from the clustering ensemble which produces the largest ANMI, and then iteratively updates the cluster label (i.e., cluster id) of each instance according to whether the change can lead to gains in ANMI, until no cluster labels will be changed for all the instances. This greedy search can significantly reduce the searching space (although it is still large), however, it tends to reach a local optimum.

5.4.2.2 Consensus function based on co-association matrix

The co-association matrix is a $n*n$ (n is the number of instances in the data set) matrix in which each element represents the association value between two instances. For each clustering, we can construct a co-association matrix and the value of each element is 1 if the two instances are clustered together, otherwise 0. To combine r clusterings, we simply sum the r co-association matrices, and then normalize the resulting matrix by dividing r for each element.

Once the final co-association matrix is obtained, we can apply any clustering algorithm on the matrix, e.g., agglomerative clustering with average-linkage. A simple demonstrative example ($r = 2$) is shown in Figure 5.2 in which the true clustering is $\{\{1,2,3\},\{4\}\}$. After we apply agglomerative clustering with average-linkage (stopping threshold is set to 0.5), we can correctly recover the true clustering from the two imperfect clusterings.

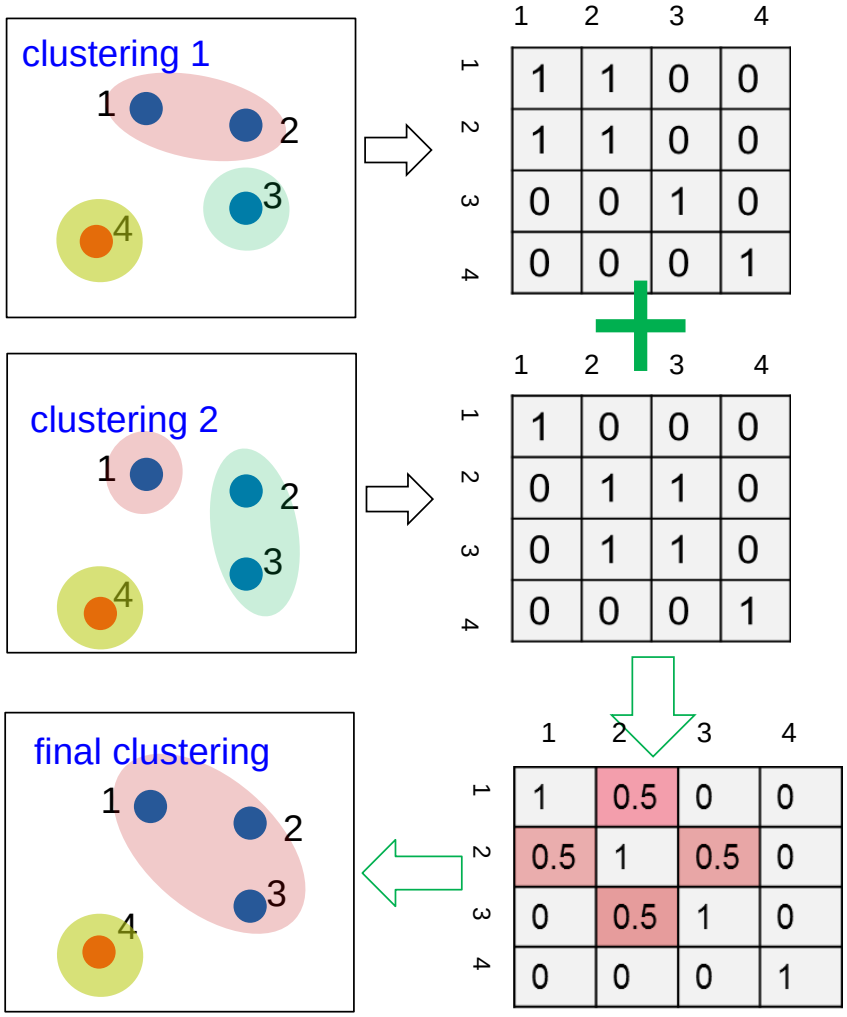


Figure 5.2: An example of consensus function based on co-association matrix

There is a major concern about the consensus function based on co-association matrix [89]: the new clustering structure represented by co-association may not reflect the original clustering structure, especially when the number of clusterers is relatively small.

5.4.2.3 Consensus function based on voting

Unlike the classification problem where a classifier produces a class label which falls in a fixed set of categories, various clusterers may produce different sets of labels. In the example shown

in Figure 5.3, clusterer 1 produces labels of {A,B,C}, clusterer 2 produces labels of {a,b,c}, while clusterer 3 produces labels of $\{\alpha, \beta, \gamma\}$. In order to apply the voting strategy, we need to relabel some of clusterings based on a single common reference clustering. Ideally, the true clustering is the best choice to be a reference clustering, unfortunately, it is not known until evaluation. Alternatively, we can use any one of the clusterings as reference clustering, as shown in the example, we use clustering 1 as the reference clustering and the labels in other clusterings are relabeled accordingly.

The relabeling problem can be converted to a bipartite graph matching problem in which labels from reference clustering form one set of nodes in the graph while labels from the other clustering form another set of nodes. There are no edges connecting nodes from the same set. The connecting edges connecting nodes in different sets show the number of common instances. The goal of bipartite graph matching is to find matches between labels such that the overall edge weights for pairwise matched labels can be maximized. In the example, we have two possible ways to relabel clustering 2: (1) $a \rightarrow A, b \rightarrow B$ in which we can obtain an overall weight of 2; (2) $a \rightarrow B, a \rightarrow A$ in which we can obtain an overall weight of 3. Therefore we choose the second relabeling solution. A general bipartite graph matching can be solved through effective Hungarian algorithm [57].

Once all the clusterings are relabeled with the same set of labels, we can simply apply majority voting for each in instance. If there is a tie, a random selection can be applied.

The major concern about the consensus functions based on voting is that the relabeling errors introduced by the bipartite graph can significantly mislead the voting.

5.4.2.4 Consensus function based on graph formulation

We can leverage graph formulation to capture the similarity of instances (IBGF), the similarity of clusters (CBGF) or both (HBGF).

(1) Instance-based graph formulation (IBGF) [85]

Given a clustering ensemble $\Pi = \{\pi^1, ..., \pi^r\}$, we can construct a fully connected graph $G = (V, W)$ in which V is a set of vertices in the graph and each vertex represents an instance; W is a similarity matrix, and the edge weight $W(i, j)$ can be computed as the frequency the

instances	true clustering	clustering 1	clustering 2	clustering 3	voting
x_1	1	A	a → B	α → B	B
x_2	2	A	b → A	β → A	A
x_3	2	A	b → A	α → B	A
x_4	1	B	b → A	α → B	B
x_5	2	A	b → A	β → A	A

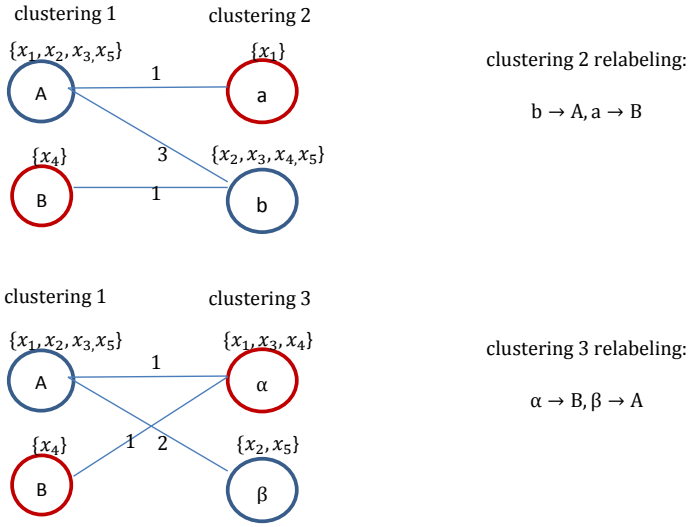


Figure 5.3: An example of consensus function based on voting

instances i and j are clustered together in the ensemble. The final clustering can be obtained by applying any graph based clustering algorithms.

(2) Cluster-based graph formulation(CBGF) [85]

Given a cluster ensemble $\Pi = \{\pi^1, ..., \pi^r\}$, we collect all the clusters from the ensemble $\Pi = \{C_1^1, ..., C_{k_1}^1, ..., C_1^r, ..., C_{k_r}^r\}$. The total number of clusters is denoted as t. Then the graph can be constructed in which V is a set of t vertices, each vertex represents a cluster; W is a matrix in which each element is the similarity between cluster c_i and c_j. One of the ways to compute the weight similarity is to apply Jaccard measure $W(i,j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}$.

Then we apply any graph clustering algorithm and each cluster contains a group of clusters in the ensemble. In order to produce the final clustering, we can consider the group of clusters as a metacluster, then we can iterate each instance in X, and count which meatacluster with which it is most frequently associated. If there are ties, we just randomly assign it to a metacluster.

(3) Hybrid bipartite graph formulation (HBGF) [33]

HBGF constructs a bipartite graph $G = (V, W)$ in which $V = V_C \cup V_I$ where V_C is a set of t vertices each representing a cluster in the ensemble as described in CBGF, V_I is a set of vertices each representing an instance.

If i and j are both clusters or instances, $W(i,j) = 0$, otherwise if instance i belongs to cluster j, $W(i,j) = W(j,i) = 1$ otherwise 0. By applying any graph-clustering algorithms, we can directly produce the final clustering.

5.5 Micro Macro Collaborative Clustering (MiMaCC)

It is quite natural to combine instance-level collaborative clustering and clusterer-level collaborative clustering so that we can first recover good clustering structure by using collaborative instances and then apply multiple clustering algorithms to produce a final clustering. The hypothesis is that clustering on data in which good structure is embedded can produce better results than clustering on data in which ill structure is embedded.

A basic algorithm to implement MiMaCC is as follows:

(1) expand the data set by introducing collaborative instances;

(2) apply clusterer-level collaborative clustering to produce a final clustering on the expanded dataset;

(3) down-scale the clustering by removing those collaborative instances from clusters.

5.6 Summary

In this chapter, we present the "collaborative clustering" scheme which includes MiCC, MaCC and MiMaCC. For MiCC, one of the key issues is to find a good internal measure and we hypothesize that if the selected instance collaborators can help improve the score of internal measure, it is also likely that we can obtain a better clustering for the original set of instances. In the next chapter, we will empirically study which internal measures are good choices, and how effective of our MiCC algorithm. For MaCC, we focus on at least two key issues, one is how to generate diverse clustering results, and the other is how to select a good consensus function. The MiMaCC algorithm is a natural extension that combines MiCC and MaCC.

Chapter 6

Document Clustering: A Case Study of Collaborative Clustering

6.1 Introduction

Document clustering is a basic clustering problem in text mining, and has been applied in several applications. For example, in information retrieval, when a document is considered as relevant to the query, then the other documents in the same cluster should also be relevant to the query. By incorporating this simple idea, recall of document retrieval can be significantly improved. For another example, the documents retrieved with respect to a query are often organized in a flat order. If the documents can be clustered into coherent groups, then the user's browsing experience can be significantly improved because instead of sifting through the long list, the user only needs to look into groups each of which represents a coherent topic/sense relevant to the query.

The research of document clustering has started decades ago, mostly focusing on comparing various clustering algorithms, e.g., [84, 100]. The observations and conclusions are based on a certain external clustering evaluation measure. However, some external measures are defective so that the conclusions may be opposite if some other measures are applied. For example, K-means algorithm tends to produce balanced cluster sizes (e.g., balanced number of documents in different clusters). If the results are evaluated by "entropy" measure, it turns out that K-

means algorithm is better than other algorithms that produce unbalanced cluster sizes. This is not desired because "entropy" can not capture the so-called "uniform effect" [92] of K-means algorithm, so the evaluation results are misleading.

In this chapter, we first study some basic issues in document clustering, including selection of *external validation measure*, selection of *distance function*, comparison among different *baseline clustering algorithms*, and then focus on experimenting the approaches proposed in Chapter 5, including instance-level collaborative learning, clusterer-level collaborative learning.

6.2 Problem Formulation

In this thesis, we follow the conventions to define the document clustering problem.

Let $\mathcal{D} = \{d_1, \ldots, d_n\}$ denote the set of n documents. Using the well-known term frequency-inverse document frequency (tf-idf) vector space model [79], a document can be represented as a vector

$$d = (w_1, ..., w_m) = (tf_1 \log \frac{n}{idf_1}, ..., tf_m \log \frac{n}{idf_m}) \tag{6.1}$$

where w_i is the tfidf value of i^{th} term, tf_i is the term frequency of i^{th} term, n is the total number of documents, and df_i is the number of documents which contain the i^{th} term. This vector is usually normalized to unit length.

The goal of document clustering is to generate a hard (non-overlapping) clustering for $\mathcal{D}$, i.e., $\mathcal{D} = (D_1, ..., D_k)$ such that

(1) $D_i \neq \emptyset$ for $i \in \{1, ..., k\}$

(2) $D_i \cap D_j = \emptyset$ for $i, j \in \{1, ..., k\}$ and $i \neq j$

(3) $D_1 \cup ... \cup D_k = \mathcal{D}$

6.3 Distance Functions

Distance functions are closely related with applications, and they are important for the success of clustering. In this chapter, we want to investigate which distance functions can be good choices for document clustering application.

A distance function normally satisfies the following three properties:

(1) symmetric, i.e., $dis(x, y) = dis(y, x)$

(2) non-negative, i.e., $dis(x, y) \geq 0$

(3) triangle inequality, i.e., $dis(x, z) \leq dis(x, y) + dis(y, z)$

We applied the following four distance functions to compute the distance between two documents:

1. Euclidean distance

This function is defined as

$$dis_{euclidean}(d_i, d_j) = \|d_i - d_j\| \tag{6.2}$$

Euclidean distance is a widely used distance function, however, the major drawback is that it is very sensitive to scaling.

2. Cosine distance

Instead of cosine distance, cosine similarity is more well-known, which can be defined as

$$sim_{cosine}(d_i, d_j) = \frac{d_i \cdot d_j}{\|d_i\| \, \|d_j\|} \tag{6.3}$$

Since we use normalized vector of d, the cosine similarity can be simplified to $d_{\cos ine} = d_i \cdot d_j$. Since the tf-idf values in a document vector are non-negative, the cosine similarity ranges from 0 to 1. 1 means that the two document vectors are in the same direction (using the same terms), 0 means that the two documents do not have any overlapping terms. Unlike Euclidean distance, cosine similarity is not sensitive to scaling because it only computes the angle between two document vectors rather than the magnitude difference. For example, considering a document containing 1 “document” and 1 “clustering” and the other document containing 100 “document” and 100 “clustering”, the cosine similarity will consider the two documents as

the same, however, the magnitude difference may be significantly different.

In order to transform cosine similarity to cosine distance, we use the following formula:

$$dis_{cosine}(d_i, d_j) = 1 - sim_{cosine}(d_i, d_j) \tag{6.4}$$

Cosine distance is non-negative, symmetric, but does not satisfy triangle inequality.

3. Distance based on Pearson Correlation Coefficient

Pearson Correlation Coefficient is a similarity measure to compute how correlated two variables are:

$$P(x, y) = \frac{n\sum xy - \sum x \sum y}{\sqrt{(n\sum x^2 - (\sum x)^2)(n\sum y^2 - (\sum y)^2)}} \tag{6.5}$$

By replacing x, y with d_i,d_j, we obtain:

$$sim_{pearson}(d_i, d_j) = \frac{n\sum_{l=1}^{m} w_{il}w_{jl} - \sum_{l=1}^{m} w_{il} \sum_{l=1}^{m} w_{jl}}{\sqrt{(n\sum_{l=1}^{m} {w_{il}}^2 - (\sum_{l=1}^{m} w_{il})^2)(n\sum_{l=1}^{m} {w_{jl}}^2 - (\sum_{l=1}^{m} w_{jl})^2)}} \tag{6.6}$$

The value of Pearson Correlation Coefficient ranges from -1 to 1, 1 means that the two documents are perfectly correlated, and -1 means that the two documents are not correlated. To transform similarity to distance, we use the following formula:

$$dis_{pearson}(d_i, d_j) = \begin{cases} 1 - sim_{pearson}(d_i, d_j) & \text{if } sim_{pearson}(d_i, d_j) >= 0 \\ |sim_{pearson}(d_i, d_j)| & \text{otherwise} \end{cases} \tag{6.7}$$

Distance based on Pearson Correlation Coefficient is non-negative, symmetric, but does not satisfy triangle inequality.

4. Distance based on Jaccard Coefficient Jaccard Coefficient is a similarity measure which computes the intersection of two sets divided by their union. For two document vectors d_i and d_j, the Jaccard coefficient can be computed as

$$sim_{Jaccard} = \frac{d_i \cdot d_j}{|d_i|^2 + |d_j|^2 - d_i \cdot d_j} \tag{6.8}$$

In order to transform Jaccard similarity to Jaccard distance, we use the following formula:

$$dis_{Jaccard}(d_i, d_j) = 1 - sim_{Jaccard}(d_i, d_j) \tag{6.9}$$

6.4 External Clustering Validation Measures

An external measure is to compare the system clustering output with the gold clustering which is normally created by human assessors. Some external measures are naturally defective, for example, using either "purity" or "entropy" measure, we can obtain the best performance by simply putting each document in a singleton cluster. Many previous researches apply those defective external measures, therefore, some of their results are misleading. In this chapter, we will investigate which external measures are good choices for our evaluation purpose.

Previously, various external measures have been surveyed in [1] and re-surveyed in [13]. The external measures can be categorized into several types as shown in Table 6.1.

Type	Representative external measures
based on set matching	Purity, F-measure
based on pair counting	Rand index, Jaccard Coefficient, Folks and Mallows
based on entropy	Entropy, Mutual Information, Variation of Information, V-measure
based on edit distance	Edit distance
mixed type	MUC F-measure, BCubed F-measure

Figure 6.1: Summary of external clustering measures by types

Researchers have proposed various properties/constraints [1, 30, 66] to identify what makes a better clustering, for example, [1] presented four formal constraints, i.e.,

(1) homogeneity: a clustering in which most of clusters contain "clean" instances is better than a clustering in which clusters are mixed with instances from various classes

(2) completeness: a clustering in which the instances belonging to the same class are put in one cluster is better than a clustering in which those instances belonging to the same class are put in different clusters

(3) rag bag: a clustering in which miscellaneous instances (belonging to different small classes) are put in one "rag bag" is better than a clustering in which those instances are dispersed in a most "clean" cluster.

(4) cluster size vs. quantity: a clustering in which instances belonging to a minority of classes are put in the same clusters is better than a clustering in which those instances are put in many fragmented clusters.

The above four constraints are demonstrated in Figure 6.2.

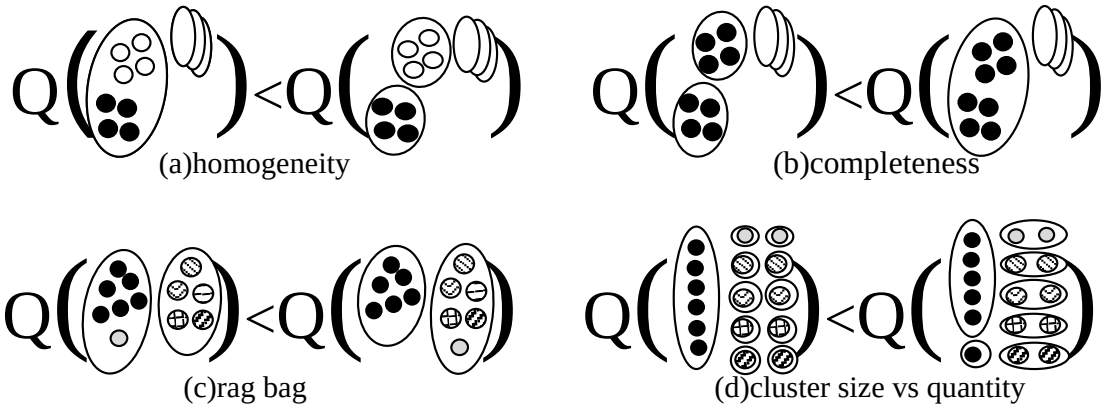


Figure 6.2: Illustrations of four formal constraints [1]

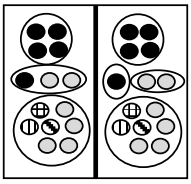
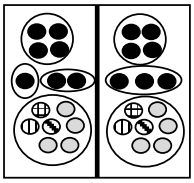
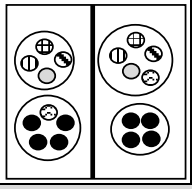
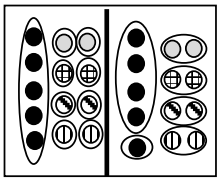
Various external measure can then be validated by checking whether they can satisfy the constraints, using the test cases as shown in Table 6.3. The table shows that only BCubed F-measure can satisfy all the four constraints, and VI, V are the second best measures which only fail the "rag bag" constraint.

Wu et al., 2009 [92] presented the "uniform effect" of K-means algorithm, and compared 16 external measures by validating whether they can capture the "uniform effect". They showed that entropy, purity, F measure, mutual information, Variation of Information (VI), Fowlkes and Mallows (FM) are defective measures which fail to capture the "uniform effect", however, the normalization forms of some measures (normalized VI, normalized FM) can well capture the "uniform effect".

In this thesis, we will focus on three external measures: normalized mutual information, BCubed and recently proposed V measure [77]. According to the four constraints discussed above, they are good choices as shown in Table 6.3, however, it is unknown whether the three measures can capture the "uniform effect" as they were not stated in [92].

1. Normalized Mutual Information

Normalized mutual information is an entropy based measure which measures the mutual dependence of two clusterings based on information theory. For the convenience of discussion, we assume that the data set consists of n data points, the system clustering is $\mathcal{C} = \{C_1, ..., C_k\}$,

Formal constraints	Homogeneity	Completeness	Rag bag	Cluster size vs. quantity
				
Measures based on set matching				
Purity	0.71 0.78 √	0.78 0.78 ×	0.55 0.55 ×	1 1 ×
Inverse purity	0.78 0.78 ×	0.78 0.78 ×	1 1 ×	0.69 0.92 √
F-measure	0.63 0.63 ×	0.62 0.62 ×	0.61 0.61 ×	0.79 0.96 √
	OK	FAIL	FAIL	OK
Measures based on pair counting				
Rand index	0.68 0.7 √	0.68 0.7 √	0.72 0.72 ×	0.95 0.95 ×
Adjusted rand*	0.25 0.28 √	0.24 0.31 √	0.4 0.4 ×	0.80 0.80 ×
Jaccard	0.31 0.32 √	0.31 0.35 √	0.37 0.37 ×	0.71 0.71 ×
F&M	0.47 0.49 √	0.47 0.52 √	0.61 0.61 ×	0.84 0.84 ×
	OK	OK	FAIL	FAIL
Measures based on entropy				
-Entropy	-1.03 -0.8 √	-0.83 -0.83 ×	-1.29 -1.29 ×	0 0 ×
Mutual Information	0.84 1.03 √	1 1 ×	0.99 0.99 ×	2.19 2.19 ×
NMI	0.50 0.58 √	0.57 0.61 √	0.61 0.61 ×	0.88 0.94 √
-VI	-1.68 -1.48 √	-1.52 -1.32 √	-1.28 -1.28 ×	-0.61 -0.27 √
V *	0.5 0.58 √	0.57 0.6 √	0.61 0.61 ×	0.88 0.94 √
	OK	OK	FAIL	OK
Measures based on edit distance				
Edit distance	7 7 ×	7 6 √	6 6 ×	9 6 √
	FAIL	OK	FAIL	OK
Mixed				
MUC F *	0.7 0.74 √	0.74 0.8 √	0.6 0.6 ×	0.67 0.93 √
B-Cubed F	0.63 0.69 √	0.66 0.67 √	0.68 0.78 √	0.78 0.89 √
ECM F *	0.57 0.57 ×	0.57 0.57 ×	0.56 0.56 ×	0.69 0.92 √
	OK	OK	OK	OK

Note: √ means satisfied, and × means not satisfied. OK means at least one of the measures in the family satisfies the constraint, FAIL means none of the measures in the family satisfy the constraint.

Figure 6.3: Satisfaction of constraints for various external measures [1], measures noted by * are extensions presented in [13]

and the reference (gold) clustering is $\Re = \{R_1, ..., R_m\}$. In the following, we use "class" to represent a cluster in reference clustering, and "cluster" to represent a cluster in system clustering.

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

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

$P(i) = |R_i|/n$: the probability of an item in class i

$P(j) = |C_j|/n$: the probability of an item in cluster j

$P(i|j) = \frac{P(i,j)}{P(j)}$: the conditional probability of an item in class i if it belongs to cluster j

$P(j|i) = \frac{P(i,j)}{P(i)}$: the conditional probability of an item in cluster j if it belongs to class i

Then, the entropy of reference clustering is

$$H(\mathcal{R}) = -\sum_{i=1}^{m} P(i) \log(P(i)) \tag{6.10}$$

The entropy of system clustering is

$$H(\mathcal{C}) = -\sum_{j=1}^{k} P(j) \log(P(j)) \tag{6.11}$$

The conditional entropy $H(\mathcal{C}|\mathcal{R})$ is

$$H(\mathcal{C}|\mathcal{R}) = \sum_{i=1}^{m} \sum_{j=1}^{k} P(i,j) \log P(j|i) \tag{6.12}$$

The conditional entropy $H(\mathcal{R}|\mathcal{C})$ is

$$H(\mathcal{R}|\mathcal{C}) = \sum_{j=1}^{k} \sum_{i=1}^{m} P(i,j) \log P(i|j) \tag{6.13}$$

The joint entropy $H(\mathcal{R}, \mathcal{C})$ is

$$H(\mathcal{R}, \mathcal{C}) = \sum_{j=1}^{k} \sum_{i=1}^{m} P(i,j) \log P(i,j) \tag{6.14}$$

Mutual Information is then defined as

$$I(\mathcal{C}; \mathcal{R}) = \sum_{i=1}^{m} \sum_{j=1}^{k} P(i,j) \cdot log \frac{P(i,j)}{P(i)\,P(j)} \tag{6.15}$$

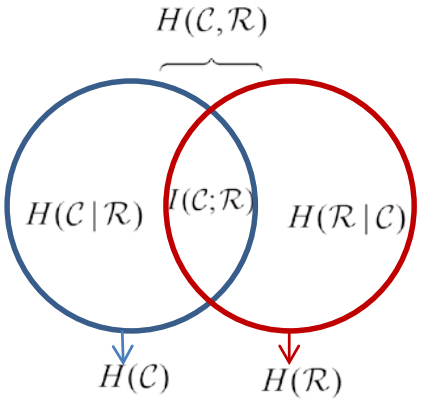
A normalized mutual information can be defined as

$$NMI(\mathcal{C}; \mathcal{R}) = \frac{2I(\mathcal{C}; \mathcal{R})}{H(\mathcal{C}) + H(\mathcal{R})} \tag{6.16}$$

The relations among entropy, conditional entropy, joint entropy and mutual information are shown in Figure 6.4 and the proves are given in [25].

2. BCubed F-measure[3]

B-Cubed measure evaluates the clustering by summing the score of each instance in the clustering.



$$I(\mathcal{C};\mathcal{R}) = H(\mathcal{C}) - H(\mathcal{C}\,|\,\mathcal{R}) = H(\mathcal{R}) - H(\mathcal{R}\,|\,\mathcal{C})$$
$$H(\mathcal{C},\mathcal{R}) = H(\mathcal{C}) + H(\mathcal{R}\,|\,\mathcal{C}) = H(\mathcal{R}) + H(\mathcal{C}\,|\,\mathcal{R})$$

Figure 6.4: Relations among entropy, conditional entropy, joint entropy and mutual information

For each instance i, the precision and recall are defined as

$$precision_i = \frac{\text{\# correct instances in the system cluster containing instance } i}{\text{\# instances in the system cluster containing instance } i} \tag{6.17}$$

$$recall_i = \frac{\text{\# correct instances in the system cluster containing instance } i}{\text{\# instances in the reference cluster containing instance } i} \tag{6.18}$$

$$precision\,(\mathcal{C}) = \sum_{i=1}^{n} w_i precision_i \tag{6.19}$$

$$recall\,(\mathcal{C}) = \sum_{i=1}^{n} w_i recall_i \tag{6.20}$$

where w_i is the weight assigned to instance i (e.g., $w_i = 1/n$).

$$BCubed\_F\,(\mathcal{C}) = \frac{2precision\,(\mathcal{C})\,recall\,(\mathcal{C})}{precision\,(\mathcal{C}) + recall\,(\mathcal{C})} \tag{6.21}$$

3. V-measure[77]

V-measure is another entropy based measure which is computed as the harmonic mean of homogeneity and completeness (refer to the beginning of this section for definitions) scores.

Homogeneity is computed as

$$h = \begin{cases} 1 & H\,(\mathcal{R}) = 0 \\ 1 - \frac{H(\mathcal{R}|\mathcal{C})}{H(\mathcal{R})} & H(\mathcal{R}) \neq 0 \end{cases} \tag{6.22}$$

Completeness is computed as

$$c = \begin{cases} 1 & H(\mathcal{C}) = 0 \\ 1 - \frac{H(\mathcal{C}|\mathcal{R})}{H(\mathcal{C})} & H(\mathcal{C}) \neq 0 \end{cases} \tag{6.23}$$

Then V-measure is computed as

$$V_\beta = \frac{(1+\beta)hc}{h+c} \tag{6.24}$$

In this thesis, we prove that normalized mutual information is equivalent to V-measure ($\beta = 1$).

Theorem 6.4.1 *Normalized mutual information is equivalent to V-measure with* $\beta = 1$.

Proof

$$\begin{aligned} V_\beta &= \frac{(1+\beta)hc}{h+c} = \frac{2(1-\frac{H(\mathcal{R}|\mathcal{C})}{H(\mathcal{R})})(1-\frac{H(\mathcal{C}|\mathcal{R})}{H(\mathcal{C})})}{(1-\frac{H(\mathcal{R}|\mathcal{C})}{H(\mathcal{R})})+(1-\frac{H(\mathcal{C}|\mathcal{R})}{H(\mathcal{C})})} \\ &= \frac{2(H(\mathcal{R})-H(\mathcal{R}|\mathcal{C}))(H(\mathcal{C})-H(\mathcal{C}|\mathcal{R}))}{H(\mathcal{C})(H(\mathcal{R})-H(\mathcal{R}|\mathcal{C}))+H(\mathcal{R})(H(\mathcal{C})-H(\mathcal{C}|\mathcal{R}))} \\ &= \frac{2I(\mathcal{C};\mathcal{R})^2}{H(\mathcal{C})I(\mathcal{C};\mathcal{R})+H(\mathcal{R})I(\mathcal{C};\mathcal{R})} \\ &= \frac{2I(\mathcal{C};\mathcal{R})}{H(\mathcal{C})+H(\mathcal{R})} \\ &= NMI \end{aligned}$$

Based on the above proof, we can reduce the three external measures to two, BCubed F-measure and V-measure ($\beta = 1$).

In order to validate whether Bcubed F-measure and V-measure can capture "uniform effect", we exactly follow the approach presented in [92]. The major math tool applied here is called the coefficient of variation which measures the skewness of cluster sizes. The larger the value is, the more skewed (unbalanced) the cluster sizes are. The coefficient of variation is computed as the ratio of the standard deviation to the mean, i.e.,

given $X = \{x_1, ..., x_n\}$,

$$CV = s/\bar{x} \tag{6.25}$$

where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$.

Let CV_0 denote the coefficient of variation for the truth clustering, and CV_1 denote the coefficient of variation for a system clustering produced by some clustering algorithm. For a

high skewed cluster distribution (CV_0 is high), an algorithm which produces "balanced" clusters normally makes CV_1 small, such that $DCV = CV_1 - CV_0$ is small. As DCV decreases, a good external measure is expected to show worse scores, so we can apply some rank correlation metric to compute the correlation between DCV and the score produced by the external measure. The higher the correlation, the better the external measure which can capture "uniform effect".

We applied two rank correlation metrics, one is Spearman's rank correlation coefficient, and the other is Kendall's rank correlation [55]. The range of either correlation coefficient falls in [-1,1], 1 means an extremely positive correlation, while -1 means an extremely negative correlation.

6.5 Experiments

6.5.1 Data Sets

We used four datasets from CLUTO package[1] which have been commonly used for evaluating clustering algorithms in previous research of document clustering. Some characteristics of the three datasets are summarized in Table 6.5. The table shows the data source (column "source"), the number of documents (column "#docs"), the number of terms (column "#terms"), the number of clusters (column "#clusters"), class distribution (in the format of "cluster label=number of documents in that cluster"), minimum cluster size in the clustering, maximum cluster size, and coefficient of variance (column "CV") [28]. Table 6.5 shows that *k1b*, *sports* have higher skewed class distribution than *reviews* and *hitech*.

The documents in the datasets have been preprocessed through the following steps:

(1)stop word removal;

(2)word stemming using Porter's stemming algorithm [72];

(3)removing any words that occur in fewer than two documents.

[1] http://glaros.dtc.umn.edu/gkhome/fetch/sw/cluto/datasets.tar.gz

dataset	source	#docs	#terms	#clusters	cluster size distribution	min cluster size	max cluster size	CV
k1b	WebACE	2340	21839	6	business=142,entertainment=1389, health=494,politics=114,sports=141, tech=60	60	1389	1.3162
reviews	San Jose Mercury (TREC)	4069	126373	5	food=999,movie=1133,music=1388, radio=137,restaurant=412	137	1388	0.6401
sports	San Jose Mercury (TREC)	8580	126373	7	baseball=3412,basketball=1410, bicycle=145,boxing=122, football=2346,golf=336,hockey=809	122	3412	1.0223
hitech	San Jose Mercury (TREC)	2301	126373	6	computer=485,electronics=116,health =603,medical=429,research=481,tech nology=187	116	603	0.5

Figure 6.5: Summary of datasets used for our experiments of document clustering

6.5.2 Experimental Tools

We developed a toolkit[1] in order to facilitate our experiments on document clustering. The toolkit consists of implementations in every aspect of clustering problem, including internal measures, external measures, distance functions, the clustering algorithms discussed in Chapter 5, instance-level collaborative clustering algorithms, and clusterer-based collaborative clustering algorithms. The toolkit also includes a wrapper for the well-known clustering toolkit, CLUTO [54] which is not open source and implemented in C++. The CLUTO toolkit implements some major clustering algorithms which are applied in our toolkits as individual clusterers in clustering ensemble.

6.5.3 Impact of External Measures

As discussed in section 6.4, BCubed F-measure, V-measure are two good external measures because BCubed F-measure satisfies all the four constraints in [1] while V-measure satisfies 3 of them. In this set of experiments, we will validate whether they can capture "uniform effect".

Following the experimental setup in [92], we used the data set *hitech* which contains 2301 documents in 6 classes: computers, electronics, health, medical, research and technology and obtained 8 sample data sets with the class-size distributions (CV0) ranging from 0.490 to 1.862, as shown in Table 6.1. For each of the 8 data sets, we repeated sampling 10 times. We applied K-means algorithm and the number of clusters (was set to the number of true classes. Table

[1] the toolkit was developed in Java programming language

6.2 shows the correlation coefficients between DCV and the four measures. We can observe that only V-measure can capture the "uniform effect", while BCubed F-measure as well as purity and entropy can not. This shows that there is not a perfect external measure discovered so far that can capture all the good constraints. However, we consider that V-measure is an ideal evaluation metric because it satisfies two very important criteria (homogeneity and completeness) and can capture "uniform effect", therefore, in the later experiments, we will only apply V-measure as our external measure.

data set	1	2	3	4	5	6	7	8
class 1	100	90	80	70	60	50	40	30
class 2	100	90	80	70	60	50	40	30
class 3	100	90	80	70	60	50	40	30
class 4	250	300	350	400	450	500	550	600
class 5	100	90	80	70	60	50	40	30
class 6	100	90	80	70	60	50	40	30
CV0	0.49	0.686	0.88	1.078	1.27	1.47	1.666	1.86

Table 6.1: 8 sample data sets from *hitech* data set

measure	Spearman's rank correlation	Kendall's rank correlation
V-measure	1.0	1.0
BCubed F-measure	-0.857	-0.714
purity	-1	-1
entropy	-1	-1

Table 6.2: Correlation between DCV and the measures

6.5.4 Impact of Similarity Functions

We experimented four similarity functions Euclidean similarity (*euc*), Cosine similarity (*cos*), Pearson Correlation Coefficient (*cor*) and Jaccard Coefficient(*jac*) on two datasets (*k1b* and *reviews*). We applied two clustering algorithms, repeated bisectional partitional clustering optimizing I_2 (*rbr+i2*) and repeated bisectional partitional clustering optimizing G_1 (*rbr+g1*). For each of the two datasets, we generated 100 datasets each containing 100 documents by random sampling such that the final result is averaged from the 100 results. For clustering, the number of clusters was set to the number of true classes, and V-measured was applied for evaluation. Wilcoxon Matched-Pairs Signed-Ranks Test was applied to validate the statistical significance.

Table 6.3 shows that among the four similarity functions, Euclidean similarity (*euc*) performs the worst except for the case when *rbr+i2* was applied on dataset *k1b*. Cosine distance

	rbr+i2				*rbr+g1*			
	cos	*euc*	*cor*	*jac*	*cos*	*euc*	*cor*	*jac*
k1b	**0.556**	0.546	0.555	0.530	0.619	0.323	**0.645**	0.630
reviews	**0.537**	0.465	0.533	0.508	**0.522**	0.232	0.517	0.470

Table 6.3: Impact of four distance functions (clustering evaluated by V-measure)

(*cos*) normally performs the best (shown as bold in the table), and *cor* almost performs as well as *cos* and we did not observe that one is statistically significant than the other at the $p < 0.05$ level except that when *rbr+g1* was applied on dataset *k1b*, *cor* can even perform better than *cos* . *jac* normally performs worse than either *cos* or *cor* with statistical significance at the $p < 0.05$ level.

Based on the above observations, we will use cosine similarity as our similarity function in the experiments of studying baseline clustering algorithms.

6.5.5 Impact of Baseline Clustering algorithms

dataset	*Agglomerative Clustering*								*Partitional Clustering*											
	linkage			*optimizing internal measure*					*repeated bisection*						*direct k-way*					
	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
k1b	0.104	0.453	**0.607**	0.477	0.451	0.398	**0.539**	0.437	0.425	**0.552**	**0.553**	0.521	**0.614**	0.554	0.530	0.544	0.545	0.521	**0.590**	0.544	0.529
reviews	0.075	**0.349**	0.275	0.326	**0.390**	0.330	0.336	0.366	0.357	0.398	**0.544**	0.497	0.520	0.531	0.514	0.415	**0.537**	0.513	0.477	0.525	0.514
sports	0.175	0.391	**0.409**	0.455	0.479	0.423	**0.506**	0.461	0.457	0.498	0.592	0.565	**0.666**	0.576	0.576	0.490	0.577	0.545	**0.597**	0.555	0.558
hitech	0.096	**0.248**	0.177	**0.266**	0.265	0.231	0.261	0.252	0.248	0.293	0.335	0.315	**0.344**	0.328	0.326	0.293	0.322	0.311	**0.330**	0.319	0.323

Table 6.4: Performance of 21 baseline clustering algorithms evaluated by V-measure

	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
slink	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
clink	>>	-	<<	<<	>	>>	<<	>>	>>	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
alink	>>	>>	-	>>	>>	>>	>>	>>	>>	>>	>>	>>	<	>>	>>	>>	>>	>>	>>	>>	>>
I_1	>>	>>	<<	-	>>	>>	<<	>>	>>	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
I_2	>>	<	<<	<<	-	>>	<<	>	>>	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
$\mathcal{E}_1$	>>	<<	<<	<<	<<	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
G_1	>>	>>	<<	>>	>>	>>	-	>>	>>	<	<	>>	<<	<	>	<	<	>	<<	<	>
H_1	>>	<<	<<	<<	<	>>	<<	-	>	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
H_2	>>	<<	<<	<<	<<	>>	<<	<	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
rI_1	>>	>>	<<	>>	>>	>>	>	>>	>>	-	>	>>	<<	>	>>	>	>	>>	<<	>	>>
rI_2	>>	>>	<<	>>	>>	>>	>	>>	>>	<	-	>>	<<	<	>>	>	>>	>>	<<	>	>>
$r\mathcal{E}_1$	>>	>>	<<	>>	>>	>>	<<	>>	>>	<<	<<	-	<<	<<	<	<<	<<	>	<<	<<	<
rG_1	>>	>>	>	>>	>>	>>	>>	>>	>>	>>	>>	>>	-	>>	>>	>>	>>	>>	>>	>>	>>
rH_1	>>	>>	<<	>>	>>	>>	>	>>	>>	<	>	>>	<<	-	>>	>	>>	>>	<<	>>	>>
rH_2	>>	>>	<<	>>	>>	>>	<	>>	>>	<<	<<	>	<<	<<	-	<<	<<	>>	<<	<<	<
dI_1	>>	>>	<<	>>	>>	>>	>	>>	>>	<	<	>>	<<	<	>>	-	>	>>	<<	>	>>
dI_2	>>	>>	<<	>>	>>	>>	>	>>	>>	<	<<	>>	<<	<<	>>	<	-	>>	<<	>	>>
$d\mathcal{E}_1$	>>	>>	<<	>>	>>	>>	<	>>	>>	<<	<<	<	<<	<<	<<	<<	<<	-	<<	<<	<<
dG_1	>>	>>	<<	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	>>	>>	>>	>>	-	>>	>>
dH_1	>>	>>	<<	>>	>>	>>	>	>>	>>	<	<	>>	<<	<<	>>	<	<	>>	<<	-	>>
dH_2	>>	>>	<<	>>	>>	>>	<	>>	>>	<<	<<	>	<<	<<	>	<<	<<	>>	<<	<<	-

Table 6.5: Significance matrix among 21 clustering algorithms on dataset *k1b*

We tested 21 baseline clustering algorithms including 9 agglomerative clustering algorithms (3 based on linkages and the other 6 each of which optimizes a specific internal measure), 12

	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
slink	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
clink	>>	-	>>	>	<<	>	>	<	<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
alink	>>	<<	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
I_1	>>	<	>>	-	<<	<	<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
I_2	>>	>>	>>	>>	-	>>	>>	>>	>>	<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
$\mathcal{E}_1$	>>	<	>>	>	<<	-	<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
G_1	>>	<	>>	>	<<	>	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
H_1	>>	>	>>	>>	<<	>>	>>	-	>	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
H_2	>>	>	>>	>>	<<	>>	>>	<	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
rI_1	>>	>>	>>	>>	>	>>	>>	>>	>>	-	<<	<<	<<	<<	<<	<	<<	<<	<<	<<	<<
rI_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	-	>>	>	>>	>>	>>	>	>>	>>	>>	>>
$r\mathcal{E}_1$	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	-	<<	<<	<<	>>	<<	<<	>	<<	<<
rG_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<	>>	-	<	>	>>	<	>	>>	<	>
rH_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	>	-	>>	>>	<	>>	>>	>	>>
rH_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	<	<<	-	>>	<<	>	>>	<	<
dI_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>	<<	<<	<<	<<	<<	-	<<	<<	<<	<<	<<
dI_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<	>>	>	>	>>	>>	-	>>	>>	>>	>>
$d\mathcal{E}_1$	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	<	<<	<	>>	<<	-	>>	<<	<
dG_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	<	<<	<<	<<	>>	<<	<<	-	<<	<<
dH_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	>	<	>	>>	<<	>>	>>	-	>>
dH_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	<	<<	>	>>	<<	>	>>	<<	-

Table 6.6: Significance matrix among 21 clustering algorithms on dataset *reviews*

	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
slink	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
clink	>>	-	<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
alink	>>	>	-	<<	<<	<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
I_1	>>	>>	>>	-	<<	>>	<<	<	<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
I_2	>>	>>	>>	>>	-	>>	<<	>>	>>	<<	<<	<<	<<	<<	<<	<	<<	<<	<<	<<	<<
$\mathcal{E}_1$	>>	>>	>	<<	<<	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
G_1	>>	>>	>>	>>	>>	>>	-	>>	>>	>	<<	<<	<<	<<	<<	>	<<	<<	<<	<<	<<
H_1	>>	>>	>>	>	<<	>>	<<	-	>	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
H_2	>>	>>	>>	>	<<	>>	<<	<	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
rI_1	>>	>>	>>	>>	>>	>>	<	>>	>>	-	<<	<<	<<	<<	<<	>>	<<	<<	<<	<<	<<
rI_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	-	>>	<<	>>	>>	>>	>>	>>	<	>>	>>
$r\mathcal{E}_1$	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	-	<<	<<	<<	>>	<	>>	<<	>	>
rG_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	-	>>	>>	>>	>>	>>	>>	>>	>>
rH_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	<<	-	<	>>	<	>>	<<	>>	>>
rH_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	<<	<	-	>>	<	>>	<<	>>	>>
dI_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	<<	<<	<<	-	<<	<<	<<	<<	<<
dI_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	<<	<<	<<	>>	-	>>	<<	>>	>>
$d\mathcal{E}_1$	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	<<	<<	<<	<<	>>	<<	-	<<	<<	<<
dG_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>	>>	<<	>>	>>	>>	>>	>>	-	>>	>>
dH_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	<	<<	<<	<<	>>	<<	>>	<<	-	<
dH_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	<	<<	<<	<<	>>	<<	>>	<<	>	-

Table 6.7: Significance matrix among 21 clustering algorithms on dataset *sports*

	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
slink	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
clink	>>	-	>>	<<	<<	>>	<	<	<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
alink	>>	<<	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
I_1	>>	>>	>>	-	>	>>	>	>>	>>	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
I_2	>>	>>	>>	<	-	>>	>	>>	>>	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
$\mathcal{E}_1$	>>	<<	>>	<<	<<	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
G_1	>>	>	>>	<	<	>>	-	>	>>	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
H_1	>>	>	>>	<<	<<	>>	<	-	>	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
H_2	>>	>	>>	<<	<<	>>	<<	<	-	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
rI_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	-	<<	<<	<<	<<	<<	>	<<	<<	<<	<<	<<
rI_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	-	>>	<	>>	>>	>>	>>	>>	>	>>	>>
$r\mathcal{E}_1$	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	-	<<	<<	<<	>>	<	>	<<	<	<
rG_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>	>>	-	>>	>>	>>	>>	>>	>>	>>	>>
rH_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	<<	-	>	>>	>	>>	<	>>	>
rH_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>>	<<	<	-	>>	>	>>	<	>	>
dI_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	<	<<	<<	<<	<<	<<	-	<<	<<	<<	<<	<<
dI_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>	<<	<	<	>>	-	>>	<	>	<
$d\mathcal{E}_1$	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	<	<<	<<	<<	>>	<<	-	<<	<	<<
dG_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<	>>	<<	>	>	>>	>	>>	-	>	>
dH_1	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>	<<	<<	<	>>	<	>>	<	-	<
dH_2	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	<<	>	<<	<	<	>>	>	>>	<	>	-

Table 6.8: Significance matrix among 21 clustering algorithms on dataset *hitech*

partitional clustering algorithms (6 repeated bisectional clustering algorithms each of which optimizes a specific internal measure and 6 k-way direct clustering algorithms each of which also

optimizes a specific internal measure). We put letter "r" or "d" in front of the internal measure name, for example, "rI_1" means the repeated bisectional clustering which optimizes I_1 measure, "dI_1" means the direct k-way clustering which optimizes I_1 measure. We experimented on the four datasets as mentioned in section 6.5.1. For each of the dataset, we generated 100 sampled datasets each of which contains 100 randomly selected documents. Table 6.4 shows the performance for 21 baseline clustering algorithms. Each number in the table shows the average V score computed by V-measure over the 100 sampled datasets in each dataset. We applied Wilcoxon Matched-Pairs Signed-Ranks test for significance test.

We compared our results with the results published in previous work [99, 100, 101]. Our experiment settings are a little different from [101] and [100], for example, we used 3 datasets and for each of them we created 100 subsets each of which contains 100 documents while in [101] they used 11 datasets (including 2 of our 3 datasets) and for each of them, they created 10 subsets each of which contains 70% of the documents from the original dataset; in [100] they used 15 datasets without sampling. The evaluation methods are also different: in [101], they applied F-score and entropy over the whole hierarchical tree (without indicating the specific clustering solution); in [100], they used entropy on each clustering with fixed k (k=5,10,15, 20) number of clusters; while in our experiments, we applied V-measure on each clustering with the number of clusters set to be the true number of classes. For statistical significant test, [101] and [100] applied paired-t test while we applied Wilcoxon Matched-Pairs Signed-Ranks test. Table 6.5,6.6,6.7 and 6.8 show the significance matrix among the 21 methods on the four datasets respectively. Note that "$\ll$" ("$\gg$") indicates that the method of the row performs significantly worse (better) than the method of the column, and "$<$" ("$>$") indicates the relationship is not significant. For all statistical significance tests, p-value is set to 0.05.

We summarize the comparisons of various observations among [100, 101] and ours in Table 6.9. Some of the observations in previous work are the same as ours. However, when comes to the comparison of methods which optimize 6 internal measures, we observed some different results. By taking a parallel comparison among the three tables Table 6.4 (our results evaluated by V-measure), Table 6.10 (results evaluated by F-score in [99][1]) and Table 6.11 (re-

[1] results for each specific dataset were not reported in [100] but in [99]

sults evaluated by entropy in [99]), we can observe that in previous work, G_1 and I_1 normally perform worse than the others and ε_1 normally has a mediocre performance, however, in our experiments, G_1 usually outperforms the others, while ε_1 and I_1 have worse performance.

We briefly go over the discussions about the common observations among [100, 101] and ours. For example, partitional clustering methods generally perform better than agglomerative clustering methods because agglomerative methods have a natural drawback in that they can not remedy the clustering mistakes which were made in the earlier stage of merging while for partitional clustering methods, the instances can be freely re-arranged in each iteration of partitioning whenever such moves lead to the optimal value computed by an internal measure. The discussions of repeated bisectional methods vs. direct k-way methods are presented in [100].

In the following, we focus on discussing the reasons that lead to the different observations between ours and [100, 101]. As discussed in Section 6.4, some clustering methods (e.g., K-means) tend to produce balanced number of clusters which is called "uniform effect". It is a desired property if the original dataset has a balanced class distribution, however, it is not desired if the original dataset has a skewed class distribution. In order to test to what extend a clustering method can capture "uniform effect", we computed $DCV = CV_1 - CV_0$ in which CV_0 denotes the coefficient of variation for the truth clustering, and CV_1 denotes the coefficient of variation for a system clustering produced by a clustering algorithm. The higher the CV_0 is, the more unbalanced class distribution in the true clustering; the higher the DCV is, the more skewed cluster distribution in the clustering generated by some clustering algorithm. Table 6.12 shows the average DCV values (except the second column CV_0) for 21 clustering methods on the 4 datasets. We can observe that $d\varepsilon_1$ and *slink* represent two extreme cases in which $d\varepsilon_1$ produces the most balanced cluster distribution while *slink* produces the most skewed cluster distribution. Besides $d\varepsilon_1$, I_2, H_1 and H_2 also produce balanced cluster distribution, while I_1, G_1, $alink$ and $clink$ produce skewed cluster distribution. By looking into the performance table in Table 6.4, both $d\varepsilon_1$ and *slink* lead to inferior performance.

Recall that we use V-measure as our evaluation metric which is computed as the harmonic mean of homogeneity and completeness. A clustering algorithm which produces balanced clus-

Observations	Zhao2004	Zhao2005	Ours
In general, repeated bisectional methods outperforms agglomerative methods	not tested	supported	supported, not only repeated bisectional clustering but also k-way direct clustering outperform agglomerative clustering
Among the three linkage based methods, slink performs the worst, and alink performs the best	not tested	supported	partially supported, slink performs the worst, but alink does not always perform the best, clink sometimes perform the best (refer to dataset *reviews and hitech*)
Among the 9 agglomerative methods, alink performs the best	not tested	supported	not supported, except k1b dataset, in the other 3 datasets, others can perform better
Among the 6 internal measure based methods, I_2 normally performs the best, H_1, H_2, ε_1, G_1 the second, and I_1 the worst	supported	supported	G_1normally performs the best, I_2, H_1, H_2 the second, ε_1 and I_1 the worst
Repeated bisectional methods outperform k-way direct methods	supported when k goes large (e.g., k>10)	not tested	supported (k is set to the number of classes)

Table 6.9: Comparisons of observations among [100, 101] and ours.

dataset	*Agglomerative Clustering*						*Partitional Clustering*					
	optimizing internal measure						*repeated bisection*					
	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2
k1b	0.836	**0.896**	0.816	0.844	0.889	0.858	0.873	0.873	0.894	0.868	0.885	**0.920**
reviews	0.642	0.689	0.690	0.654	0.648	**0.727**	0.684	0.822	**0.866**	0.790	0.753	0.762
hitech	0.480	0.480	**0.489**	0.471	0.476	0.468	0.473	0.556	0.571	0.522	0.545	**0.580**

Table 6.10: Performance of 12 clustering methods (k=10, 10-way clustering) on three datasets evaluated by F-score, reported in [99].

dataset	*Agglomerative Clustering*						*Partitional Clustering*					
	optimizing internal measure						*repeated bisection*					
	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2
k1b	0.276	0.180	0.280	0.174	**0.173**	0.235	0.187	0.172	0.152	0.156	0.183	**0.133**
reviews	0.606	0.442	0.460	0.506	0.436	**0.402**	0.359	0.299	**0.232**	0.288	0.316	0.254
hitech	0.734	0.726	0.731	0.714	0.720	**0.710**	0.666	0.583	0.575	0.602	0.593	**0.571**

Table 6.11: Performance of 12 clustering methods (k=10) on three datasets evaluated by entropy, reported in [99].

dataset	CV_0	*Agglomerative Clustering*									*Partitional Clustering*											
		linkage			*optimizing internal measure*						*repeated bisection*						*direct k-way*					
		slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
k1b	1.307	**0.973**	-0.798	-0.052	0.160	-1.050	-1.177	-0.080	-0.912	-1.150	-0.126	-1.116	-1.184	-0.510	-1.038	-1.172	-0.309	-1.115	**-1.205**	-0.636	-1.067	-1.187
reviews	0.665	**1.452**	0.029	0.960	0.730	-0.325	-0.330	0.768	-0.236	-0.336	0.481	-0.468	-0.582	0.171	-0.385	-0.565	0.345	-0.484	**-0.586**	0.065	-0.411	-0.568
sports	0.980	**1.451**	-0.267	0.787	0.182	-0.733	-0.772	0.410	-0.676	-0.773	-0.031	-0.804	-0.897	-0.102	-0.757	-0.872	-0.105	-0.796	**-0.898**	-0.253	-0.773	-0.873
hitech	0.534	**1.760**	0.160	1.376	1.009	-0.302	-0.404	0.979	-0.182	-0.373	0.736	-0.389	-0.451	0.388	-0.308	-0.442	0.576	-0.395	**-0.472**	0.248	-0.333	-0.453

Table 6.12: DCV of 21 baseline clustering algorithms

ter distribution tends to split a large cluster into smaller clusters. The splitting has two potential effects: in one hand, it can potentially improve the homogeneity if the large cluster contains instances from mixed classes and splitting leads to more cleaner clusters; in the other hand, it

can potentially hurt the completeness if the large cluster already contains most clean instances from one class and splitting leads to more dispersed clusters in which the instances are from the same class. We pick out one of the 100 sampled datasets in the k1b dataset, and illustrate the cluster distribution produced by *slink,clink,alink*, dI_1, dI_2, $d\varepsilon_1$, dG_1, dH_1, and dH_2 in Table 6.13. We can observe that $d\varepsilon_1$, I_2, H_1 and H_2 which have been shown to produce balanced cluster distribution in Table 6.12 lead to relative higher homogeneity and lower completeness compared with I_1, $alink$, $clink$ which produce skewed cluster distribution. By looking into more sampled datasets with relative high skewed class distribution (e.g., sampled datasets in k1b dataset with average $CV_0 = 1.307$), we observe the follows that were not stated in previous work:

1. $alink$ and G_1 consistently perform better than $d\varepsilon_1$, I_2, H_1 and H_2 with statistical significance, and I_1 tend to perform better without statistical significance. As shown in the example in Table 6.13, $alink$'s clustering contains several small and tight clusters and a cluster which consist of mixed types of documents (cid=5), G_1 also contains several tight clusters and a relative small cluster with mixed types of documents. Furthermore, the high skewed class "heal" is mainly distributed in cid=5 in $alink$ and cid=5 in G_1 while it is distributed in 3 to 4 clusters in $d\varepsilon_1$, I_2, H_1 and H_2 which leads to much worse completeness.

2. $slink$ performs the worst because it produces several singleton clusters and one very large and loose cluster with mixed types of documents.

As the dataset becomes more balanced (e.g., sampled datasets in dataset of *reviews* or *hitech*), I_2, H_1 and H_2 start to supersede I_1, *alink* and G_1 which become better choices, however, rG_1 still remains the best choice for the dataset of *hitech* as shown in Table 6.4.

In conclusion, some of observations in previous work are contradictory with ours because the evaluation metric (e.g., purity, entropy) applied in their experiments favors those clustering algorithms which tend to generate balanced cluster distribution while we applied V-measure which can capture the "uniform effect".

cid	size	purity	heal	spor	busi	poli	ente	tech
0	1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0	0
2	1	1	1	0	0	0	0	0
3	1	1	1	0	0	0	0	0
4	1	1	1	0	0	0	0	0
5	95	0.589	56	7	18	7	1	6

(a) Clustering by *slink* (V=0.0346, h=0.0685, c=0.2729)

cid	size	purity	heal	spor	busi	poli	ente	tech
0	23	0.304	7	6	0	3	1	6
1	29	1	29	0	0	0	0	0
2	16	0.625	10	1	1	4	0	0
3	7	1	7	0	0	0	0	0
4	17	1	0	0	17	0	0	0
5	8	1	8	0	0	0	0	0

(b) Clustering by *clink* (V= 0.4883, h= 0.5867, c= 0.4182)

cid	size	purity	heal	spor	busi	poli	ente	tech
0	17	1	0	0	17	0	0	0
1	3	1	3	0	0	0	0	0
2	1	1	1	0	0	0	0	0
3	6	1	0	0	0	0	0	6
4	2	1	2	0	0	0	0	0
5	71	0.775	55	7	1	7	1	0

(c) Clustering by *alink* (V= 0.6049, h= 0.5406, c= 0.6866)

cid	size	purity	heal	spor	busi	poli	ente	tech
0	3	1	0	3	0	0	0	0
1	5	0.6	2	0	3	0	0	0
2	7	0.714	1	1	0	5	0	0
3	14	1	0	0	14	0	0	0
4	16	0.688	11	0	0	0	0	5
5	55	0.855	47	3	1	2	1	1

(d) Clustering by dI_1 (V= 0.5213, h= 0.552, c= 0.4938)

cid	size	purity	heal	spor	busi	poli	ente	tech
0	13	0.462	2	5	0	6	0	0
1	18	1	0	0	18	0	0	0
2	16	0.875	14	1	0	1	0	0
3	16	0.625	10	0	0	0	0	6
4	18	0.889	16	1	0	0	1	0
5	19	1	19	0	0	0	0	0

(e) Clustering by dI_2 (V= 0.5427, h= 0.6757, c= 0.4534)

cid	size	purity	heal	spor	busi	poli	ente	tech
0	16	0.375	5	5	0	6	0	0
1	18	1	0	0	18	0	0	0
2	15	0.733	11	2	0	1	1	0
3	16	1	16	0	0	0	0	0
4	17	1	17	0	0	0	0	0
5	18	0.667	12	0	0	0	0	6

(f) Clustering by $d\varepsilon_1$ (V= 0.5216, h= 0.6506, c= 0.4353)

cid	size	purity	heal	spor	busi	poli	ente	tech
0	3	1	0	0	3	0	0	0
1	7	0.857	1	0	0	6	0	0
2	14	1	0	0	14	0	0	0
3	14	0.429	5	6	1	1	1	0
4	21	0.714	15	0	0	0	0	6
5	41	0.976	40	1	0	0	0	0

(g) Clustering by dG_1 (V= 0.5962, h= 0.6803, c= 0.5306)

cid	size	purity	heal	spor	busi	poli	ente	tech
0	14	0.429	3	5	0	6	0	0
1	18	1	0	0	18	0	0	0
2	15	0.8	12	2	0	1	0	0
3	17	1	17	0	0	0	0	0
4	18	0.611	11	0	0	0	1	6
5	18	1	18	0	0	0	0	0

(h) Clustering by dH_1 (V= 0.5399, h= 0.6729, c= 0.4508)

cid	size	purity	heal	spor	busi	poli	ente	tech
0	15	0.6	9	5	0	0	1	0
1	18	1	0	0	18	0	0	0
2	15	0.467	7	1	0	6	0	1
3	18	0.889	16	1	0	1	0	0
4	17	1	17	0	0	0	0	0
5	17	0.706	12	0	0	0	0	5

(i) Clustering by dH_2 (V= 0.487, h= 0.6074 , c= 0.4065)

Table 6.13: Cluster distribution generated by various clustering methods. cid: cluster id; size: number of documents in this cluster; heal,spor,busi,poli,ente,tech are 6 class names

6.5.6 Correlation between Internal Measures and V-Measure

In Chapter 5, section 5.2.2, we studied 12 internal measures. In previous work, the internal measures were mainly experimented by validating whether they can capture the correct number of classes k [68, 90]. In this thesis, we apply a specific internal measure as our objective

function in the Micro Collaborative Clustering (MiCC) algorithm, in other words, the goal of MiCC is to optimize some internal measure by iteratively adding a certain number of instance collaborators. In this set of experiments, we have two goals:

(1) identifying whether the 12 internal measures can be applied for determining the number of clusters;

(2) identifying which internal measure has the highest correlation with V-measure.

To achieve the first goal, we experimented with one sampled dataset from the dataset of *k1b* by applying rbr+g1 (repeated bisectional clustering which optimizes G_1). The true number of classes in *k1b* is 6. Table 6.14 shows the scores computed by the 12 internal measures[1] as the number of clusters k gradually increases. In each column, we highlight the optimal values as bold. We can see that as k increases, the scores of I_1, I_2, ε_1, G_1, H_1, H_2, *CH*, *D* and *SD* monotonically increase or decrease, therefore they are not good measures to determine the size of clusters. *SC*, *DB* and *S_Dbw* reach the optimal when $k = 5$, $k = 5$ and $k = 8$ respectively, therefore, they are good candidates to determine the best number of clusters k.

k	I_1	I_2	ε_1	G_1	H_1	H_2	CH	D	SC	DB	SD	S_Dbw
2	4.5169	20.7477	1333.513	**2.4052**	0.0034	0.0156	6298.732	**29618951**	0.6598	36.2463	**31.4155**	2.0567
3	6.1856	24.0647	1172.402	3.9221	0.0053	0.0205	2362.854	27716557	0.66	38.2832	15.1044	2.2644
4	7.7192	27.3005	1006.277	5.5865	0.0077	0.0271	1179.498	22715856	0.6223	36.5277	15.8475	2.3946
5	9.0265	29.0833	963.107	7.2696	0.0094	0.0302	786.4919	2989.165	**0.6624**	**45.7985**	6.3662	2.4627
6	10.5779	30.5881	945.7332	9.0556	0.0112	0.0323	581.2657	4546.656	0.6173	40.6919	4.6038	2.4638
7	11.6117	31.7443	928.3161	10.8472	0.0125	0.0342	453.2753	4546.656	0.5346	42.5396	2.983	2.5747
8	12.8605	33.8697	852.863	12.7716	0.0151	0.0397	330.9466	4546.656	0.5106	42.007	2.8136	**2.5755**
9	**14.0543**	**34.9775**	**839.6695**	14.7372	**0.0167**	**0.0417**	**273.068**	4546.656	0.5069	41.9235	2.6468	2.3533

Table 6.14: Impact of 12 internal measures in determining the number of clusters k

To achieve the second goal, we applied the 21 baseline clustering algorithms as experimented in section 6.5.5, and for each generated clustering, we computed the clustering quality by the 12 internal measures respectively and the V score computed by V-measure. We then apply the Spearman's rank correlation to compute the correlation between each internal measure with V-measure. The higher correlation, the better the internal measure is, in other words, if the clustering quality computed by an internal measure has a higher score, it is also likely that the score computed by V-measure has a higher score. That is the basis of our Micro Collaborative Clustering (MiCC) algorithm. Through optimizing some internal measure after adding

[1] we applied similarity function in the formula

collaborative instances, we expect to achieve a higher score computed by external measure such as V-measure. Table 6.15 shows that the correlation between each internal measure and V-measure. We can observe that for dataset *k1b* and *sports*, G_1 has a higher correlation than the other five internal measures (I_1, I_2, ε_1, H_1, H_2) that were applied as optimization functions in our baseline clustering algorithms. That further confirms that optimizing G_1 leads to better performance evaluated by V-measure on those two datasets. We can also observe for dataset *reviews*, I_2 and H_1 has a relative higher correlation compared with the other four (I_2, ε_1, H_2) and optimizing them leads to better performance as shown in Table 6.4. Among the left 6 internal measures (*CH*,*D*,*SC*,*DB*,*SD*,*S_Dbw*), we can see that *SC* consistently has the highest correlation with V-measure.

By taking the above two goals into considerations, we finally choose Silhouette Coefficient (*SC*) as our internal measure applied in MiCC algorithm because *SC* can determine the number of clusters[1] and it has high correlation with V-measure.

dataset	I_1	I_2	ε_1	G_1	H_1	H_2	CH	D	SC	DB	SD	S_Dbw
k1b	0.430	0.119	0.005	**0.642**	0.156	0.050	0.050	0.204	**0.469**	0.450	-0.337	0.287
reviews	0.182	**0.627**	0.571	0.423	**0.641**	0.592	0.592	0.403	**0.681**	0.134	0.202	0.149
sports	0.088	0.330	0.292	**0.668**	0.313	0.307	0.307	0.325	**0.764**	0.315	0.071	0.457

Table 6.15: Correlation between internal measures and V-measure

6.5.7 Impact of MiCC

We experimented on three datasets: *k1b*, *reviews* and *sports*. Similar with the experimental setup for evaluating the performance of baseline clustering algorithms, in each of the three datasets, we generated 100 sampled datasets each of which contains 100 randomly selected documents. The pool of collaborative documents contains all the left documents in the dataset excluding the 100 selected initial documents. We selected Silhouette Coefficient (*SC*) as our internal measure. For each iteration, we selected at most $n_{step} = 20$ collaborative documents. To pick the best 20 collaborative documents in each iteration, we tried $n_{trials} = 10$ times by randomly selecting 20 documents from the pool. For each trial, we added the 20 documents

[1] we do not leverage this good property in the experiments of document clustering since we set the number of clusters to be the number of classes, however, in the case study of name entity clustering, we will leverage it to determine the number of clusters

into the expanded set of documents (in the first iteration, the expanded set only contains 100 initial documents), and apply a clustering algorithm on the newly expanded set of document and then compute the clustering quality of the original 100 documents using *SC*. Among the 10 trials, we pick the best 20 collaborative documents which lead to the best clustering quality. We set $n_{iterations} = 5$, in other words, we at most select 100 collaborative documents in the end.

We tested on the 12 baseline clustering algorithms (I_1, I_2, ε_1, G_1, H_1, H_2, rI_1, rI_2, $r\varepsilon_1$, rG_1, rH_1, rH_2) , and validated whether the MiCC algorithm can help each baseline algorithm to achieve better performance evaluated by V-measure. Figure 6.6, 6.7 and 6.8 show the impact of applying MiCC on top of 12 baseline clustering algorithms on three datasets *k1b*, *reviews*, and *sports* respectively. Each number on the bar is the average V score over the 100 sampled datasets in each dataset and the Wilcoxon Matched-Pairs Signed-Ranks test was applied to test statistical significance (p-value is set to 0.05).

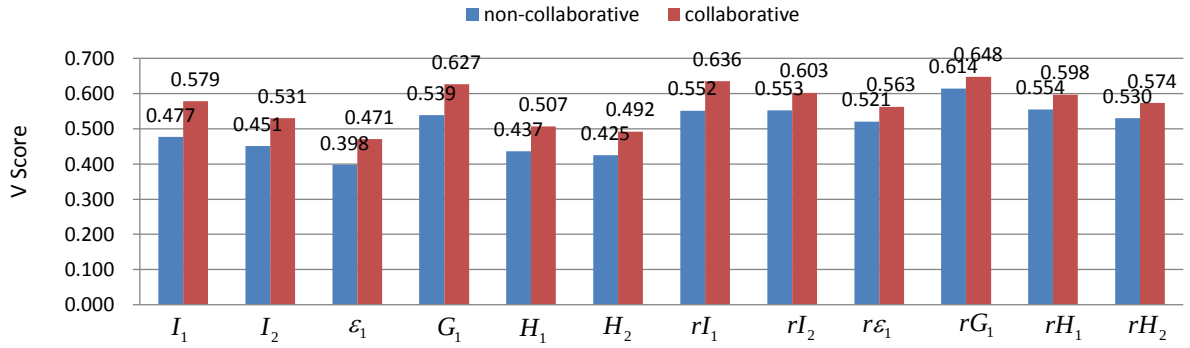


Figure 6.6: Impact of MiCC algorithm on the dataset *k1b*

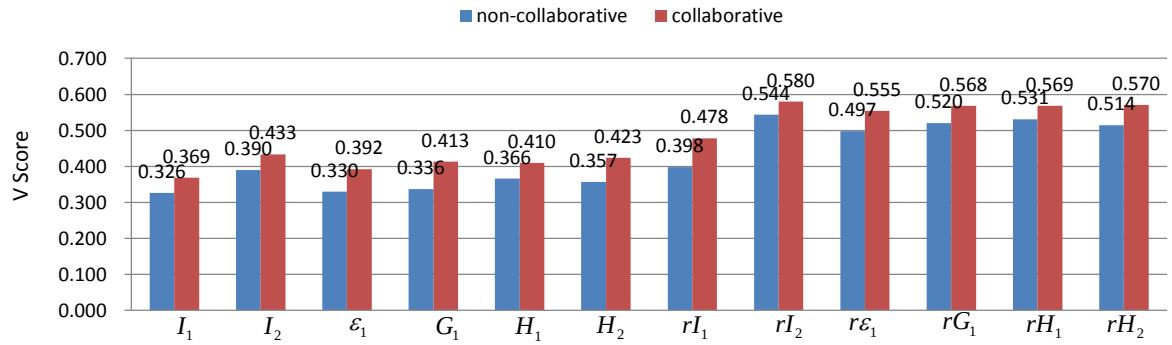


Figure 6.7: Impact of MiCC algorithm on the dataset *reviews*

From the three figures, we can observe that the MiCC algorithm on top of every baseline clustering algorithm can outperform the baseline clustering algorithm itself and all the improvements are statistical significant. For example, on dataset *k1b*, we observed 3.4% performance

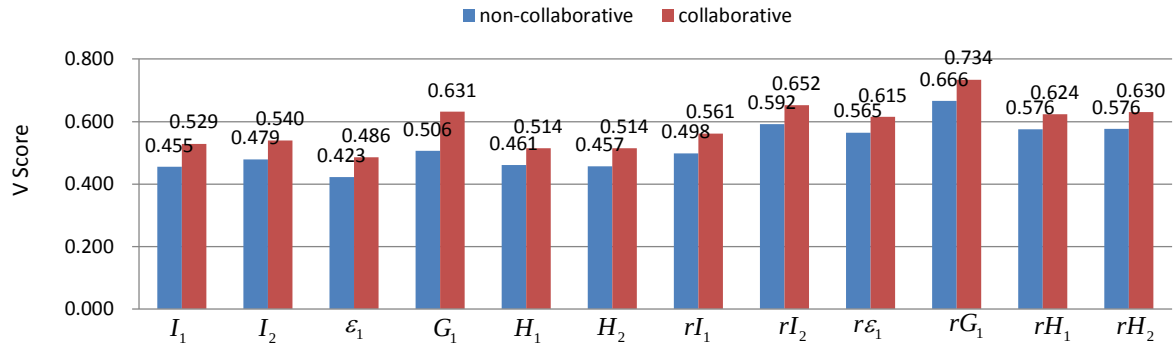


Figure 6.8: Impact of MiCC algorithm on the dataset *sports*

gains over the best baseline clustering algorithm which is rG_1, on dataset *reviews*, we observed 3.6% performance gains over the best baseline clustering algorithm which is rI_2, and on dataset *sports*, we observed 6.8% performance gains over the best baseline clustering algorithm which is rG_1.

Table 6.16 shows the average number of iterations that is needed by each baseline clustering algorithm to reach the optimum value of silhouette coefficient. We can observe that in dataset *k1b* and *sports*, rG_1 is faster than the others (with statistical significance) to reach the optimum value of silhouette coefficient. More importantly, we also know that rG_1 performs significantly better than the others. This is an important property that rG_1 can not only take shorter time to converge, but also perform well. A similar observation applies to rH_1 in dataset *reviews*.

dataset	I_1	I_2	ε_1	G_1	H_1	H_2	rI_1	rI_2	$r\varepsilon_1$	rG_1	rH_1	rH_2
k1b	1.99	1.75	1.79	1.85	2.63	1.48	2.12	2.33	1.76	**1.38**	2.2	1.79
reviews	2.31	2.11	1.58	2.09	2.61	1.8	2.2	2.8	2.22	1.86	**1.55**	2.61
sports	2.22	1.92	2.37	2.29	2.38	2.09	2.05	2.49	2.1	**1.68**	2.58	2.19

Table 6.16: Average number of iterations that is needed to optimize Silhouette Coefficient

6.5.8 Impact of MaCC

We collected 80 clustering solutions including:

(1) 24 solutions based on agglomerative clustering which is obtained by 8 agglomerative clustering algorithm[1] (clink, alink, I_1, I_2, ε_1, G_1, H_1, H_2) and 3 different similarity functions: cosine, correlation, and Jaccard.

(2) 18 solutions based on repeated bisectional partitional clustering which is obtained by

[1]because of significant bad performance of slink as showed earlier, we do not use it here

6 agglomerative clustering algorithm (rI_1, rI_2, $r\varepsilon_1$, rG_1, rH_1, rH_2) and 3 different similarity functions: cosine, correlation, and Jaccard.

(3) 18 solutions based on direct k-way partitional clustering which is obtained by 6 agglomerative clustering algorithm (dI_1, dI_2, $d\varepsilon_1$, dG_1, dH_1, dH_2) and 3 different similarity functions: cosine, correlation, and Jaccard.

(4) 20 solutions based on K-means algorithm with 20 different initial seed settings.

The goals of this set of experiments are

(1) to identify whether we can achieve even higher performance than the best baseline, if not, whether the combined clustering can outperform most of the individual clusterings. (2) to identify how the performance changes by incrementally adding more clustering solutions.

In order to achieve the above goals, we experimented four different schemes of combining the 80 clustering solutions[1]:

Scheme 1. As we have observed from the earlier experiments that clusterers using cosine similarity can outperform those using correlation similarity, and then outperform those using Jaccard similarity. Therefore, we split the 60 clusterings (24+18+18) into 3 serial groups, the first group uses cosine similarity, the second group uses correlation similarity, and the third group uses Jaccard similarity. Each group contains 20 clusterers (8 agglomerative+6 repeated bisectional+6 direct k-way). We use notations "cos", "cor" and "jac" to represent the three groups.

Scheme 2. As we have observed from the earlier experiments that repeated bisectional partitional clustering can outperform direct k-way partitional clustering in general, and then agglomerative clustering. We split all of the 80 clusterings into 4 serial groups, the first group consists of repeated bisectional partitional clustering (24 = 8×3 clusterers), the second group consists of direct k-way partitional clustering (18 = 6×3 clusterers), the third consists of agglomerative clustering (18 = 6×3 clusterers) and the fourth consists of 20 K-means clusterings. We use notations "rbr", "direct", "agglo" and "kmeans" to represent the four groups respectively.

Scheme 3. We rank the 80 clusterings by computing the internal measure "silhouette coef-

[1] for scheme 1, we only use 60 clustering solutions

ficient" from the highest to the lowest, and then split them into 4 groups, each consisting of 20 clusterings.

Scheme 4. We rank the 80 clusterings by computing the external measure "V-measure" from the highest to the lowest, and then split them into 4 groups, each consisting of 20 clusterings.

We experimented four consensus functions:

(1) Co-association matrix based (**Co.**): we first construct a matrix in which each element is computed as the frequency of the two instances are clustered together among the set of clusterings. Then we apply agglomerative clustering with average-linkage to obtain the final clustering.

(2) IBGF: we first construct a fully connected graph in which each node represent the instance, and the edge weight represents the frequency of the two instances are clustered together among the set of clusterings. Then we apply graph partitional algorithm (i.e., repeated bisectional partitional clustering which optimizes G_1) to obtain the final clustering.

(3) CBGF: similarity with IBGF, we first construct a fully connected graph, however, each node represent a cluster from the cluster ensemble of the clusterings, and the edge weight represents the linked strength of two clusters by computing the Jaccard similarity. Then we apply graph partitional algorithm (i.e., repeated bisectional partitional clustering which optimizes G_1) to obtain the clustering of those clusters. In order to obtain clustering for instances, we assign each instance to the most frequently assigned clusters.

(4) HBGF: we construct a bipartite graph, and the nodes are split into two sets, one set to represent instances, and the other set to represent clusters as in CBGF. The edge weight is 1 if an instance is included in a cluster. Then we apply graph partitional algorithm (i.e., repeated bisectional partitional clustering which optimizes G_1) to obtain the final clustering.

We experimented on three datasets: *k1b*, *reviews* and *sports*. Similar with the settings in testing baseline clustering algorithms, for each of the dataset, we reused the 100 sampled datasets each of which contains 100 randomly selected documents. We computed average *V* score over the 100 sampled datasets in each dataset. We applied Wilcoxon Matched-Pairs Signed-Ranks test for significance test.

Table 6.17 summarizes the statistics from our experiments. The columns of "max", "min", "ave", "std" represent the maximum V-score, minimum V-score, average V-score, standard deviation of the individual clustering solutions in the ensemble. The columns of "macc-similarity", "macc-algorithm", "macc-internal", "macc-external" represent the four incremental combination schemes as above. "best" represents the best consensus function that we can obtain the best result, "score" shows the V-score by applying the best consensus function, and "rank(%)" represent the rank position of the combined score among the V-score from the individual clustering solutions, for example, "100%" represents the combined solution can outperform all the individual clustering solutions, and "97.5%" represents the combined solution can outperform 97.5% of all the individual clustering solutions.

The separated results of the four combination schemes are shown in Figure 6.9, 6.10, 6.11 and 6.12 respectively.

By analyzing Figure 6.9 and Table 6.17, we can observe that:

(1) For dataset *k1b*, the consensus function "CBGF" outperforms the others with statistical significance, and for dataset *reviews* and *sports*, co-association matrix based (**Co.**) performs the best with statistical significance.

(2) The best combined scores for dataset *k1b* and *sports* can not beat the best individual clustering solution, however, it beats 97.5% of the clusterers. The best combined score for dataset *reviews* can beat all the individual clusterers.

(3) For dataset *k1b*, there is not a significant performance improvement or drop as we incrementally add more clustering solutions with different similarity functions. The reason could be that the performance of clusterers using cosine similarity is comparable with the performance of clusterers, and then comparable with the performance of clusterers, as we computed the average V scores for the three groups of clusters(group 1: clusterers with cosine similarity, group 2: clusterers with correlation similarity, group 3: clusterers with Jaccard similarity), which are 0.522, 0.519 and 0.496 respectively. Therefore, the change of similarity functions does not provide great variation in clustering results. Diversity is a key issue for the success of clustering ensemble (MaCC).

By analyzing Figure 6.10 and Table 6.17, we can observe that:

dataset	max	min	ave	std	macc-similarity			macc-algorithm			macc-internal			macc-external		
					best	score	rank(%)	best	score	rank(%)	best	score	rank(%)	best	score	rank(%)
k1b	0.645	0.364	0.516	0.056	CBGF	0.620	97.5	CBGF	0.618	97.5	CBGF	0.645	100	CBGF	**0.693**	100
reviews	0.544	0.275	0.462	0.070	Co.	0.558	100	Co.	0.567	100	Co.	0.572	100	Co.	**0.602**	100
sports	0.666	0.235	0.514	0.089	Co.	0.627	97.5	Co.	0.626	97.5	Co.	0.651	98.8	Co.	**0.680**	100

Table 6.17: Summary of MaCC experiments

(1) For dataset *k1b*, the performance can be significantly improved by incrementally adding clusterers which are from different algorithm genres, for example, using "co-association matrix" and "CBGF", the performance gains are 2.6% (from 0.582 to 0.608) and 2.5% (from 0.593 to 0.618) respectively. However, the best combined V score 0.618 which is obtained by "CBGF" still can not beat the best individual score which is 0.645.

(2) For dataset *reviews*, we can observe a significant drop after we add "agglo" clusterers. We computed the average V score for each series of clusterers (rbr, direct, agglo, kmeans), which are 0.498, 0.488, 0.366, and 0.519 respectively. As the average performance of agglo series of clusterers is significantly worse than the other three series, therefore, the worse clusterings from agglo series of clusterers weaken the combined clustering after they are added, however, we can observe significant improvement after we add kmeans series of clusterers, as they perform well individually. That tells us that good individual clusterers (collaborators) can bootstrap the combined performance, while bad individual clusterers can hurt the group strength.

(3) For dataset *sports*, we found an interesting problem: we still observed a drop by applying IBGF, CBGF or HBGF, since the average performance of agglo series of clusterers is also much worse than the others, however, by applying co-association matrix, we observed an unexpected performance increase. We believe that the main reason is that the clustering method applied in co-association matrix is more effective than the others to take advantage of group combination. That may tell us that consensus function is also important for success of MaCC. In fact, by comparing among the few figures, we can observe that co-association matrix method normally works well, although it is also simple.

By analyzing Figure 6.11 and Table 6.17, we can observe that:

(1) For dataset *k1b* and dataset *reviews*, after we add the top 40 clusterers (ranked by silhouette coefficient), we can obtain the best performance by applying CBGF on *k1b* and apply-

ing co-association matrix on *reviews* respectively. The best combined V score on *k1b* is very slightly higher than the best individual V score without statistical significance.

(2) For dataset *sports*, the best performance is achieved when we only add the top 20 clusterers.

(3) most importantly, we can observe this combination scheme can achieve better result than the previous two combination schemes. That can tell that selection of good clusterers (collaborators) is important for the success of MaCC.

We now look at the Figure 6.12, and we can observe that:

(1) For all the three datasets, the best combined V scores outperform the best individual V scores. On dataset *k1b*, it outperforms the best by 4.8% (0.693 vs. 0.645) and outperforms the average by 17.7% (0.693 vs. 0.516). On dataset *reviews*, it outperforms the best by 5.8% (0.602 vs. 0.544) and outperforms the average by 14% (0.602 vs. 0.462). On dataset *sports*, it outperforms the best by 1.4% (0.680 vs. 0.666) and outperforms the average by 16.6% (0.680 vs. 0.666).

(2) For all the three datasets, we can observe very consistent tendency as we incrementally add 20 more clusterers from higher performance to lower performance. The combined V score all achieve the best when only the top 20 clusterers are added. That can tell that the elite collaborators in fact produce even better results, while after adding inferior collaborators, the performance drops.

However, this scheme is not realistic for every case, since we leverage the gold clustering and external measure to rank the clusterers.

To summarize this set of experiments, we come to the following conclusions:

(1) the success of MaCC relies on clustering diversity. However, such diversity is based on the precondition that the individual performance is reasonably good.

(2) the success of MaCC relies on selection of clusterers. Good clusterers tend to produce even better final result, while bad clusterers tend to produce negative result.

(3) the success of MaCC also relies on consensus function. Co-association matrix based and CBGF are normally two good choices according to our observations on the three datasets.

6.5.9 Impact of MiMaCC

For this set of experiments, we first apply MiCC algorithm which uses rG_1 as the baseline clustering algorithms to obtain a best set of instance collaborators, then we apply the MaCC algorithm which uses the combination scheme of ranking 80 clusterers by V measure to obtain a combined clustering on the expanded set. Finally, we down-scale the clustering by removing those collaborative instances.

Figure 6.13 shows that using MiMaCC, we can achieve even much better result than either MiCC or MaCC. By parallelly comparing Figure 6.13, Figure 6.6, Figure 6.7, Figure 6.8 and Table 6.17, we can observe that on dataset *k1b*, it outperforms the best MiCC by 8.5% (0.733 vs. 0.648) and outperforms the best MaCC by 4% (0.733 vs. 0.693). On dataset *reviews*, it outperforms the best by 3.4% (0.614 vs. 0.580) and outperforms the best MaCC by 1.2% (0.614 vs. 0.602). On dataset *sports*, it is slightly worse than the best MiCC by 0.1% (0.733 vs. 0.734) and outperforms the best MaCC by 5.3% (0.733 vs. 0.680).

6.6 Summary

In this chapter,

(1) we have compared various external measures and picked V-measure as our evaluation metric, because it captures the most important characteristics in measuring the quality of clusterings: homogeneity, completeness and uniform effect.

(2) we have compared four similarity functions and picked cosine similarity as our similarity function in document clustering, because of its stable and good performance across datasets.

(3) we have compared 21 baseline clustering algorithms, and identified that for dataset with unbalanced class distribution, rG_1 may be the best choice since it does not intend to produce balanced clusters, while for dataset with balanced class distribution, rI_2 may be a better choice since it tends to produced very balanced clusters.

(4) we have compared 12 internal measures and picked Silhouette Coefficient as our internal measure that is further applied in MiCC algorithm because Silhouette Coefficient can not only capture the size of clusters, but also has the highest correlation with V-measure which indicates

that if a clustering has a high Silhouette Coefficient, the clustering is also likely to be a good clustering with a high V score.

(5) we have studied the impact of MiCC by using various baseline clustering algorithms, and showed that collaborative instances are really helpful to reconstruct the clustering structure if the structure in the original dataset is vague and not easy for any baseline clustering algorithm to identify.

(6) we have studied the impact of MaCC using four different combination schemes, and identified three factors that can affect the success of MaCC: diversity, selection scheme, and consensus function.

(7) we have showed that MiMaCC produces the best results we have achieved so far.

Figure 6.9: Impact of MaCC by incrementally adding clusterers using 3 serial similarity functions: cosine similarity (cos), correlation similarity (cor) and Jaccard similarity (jac)

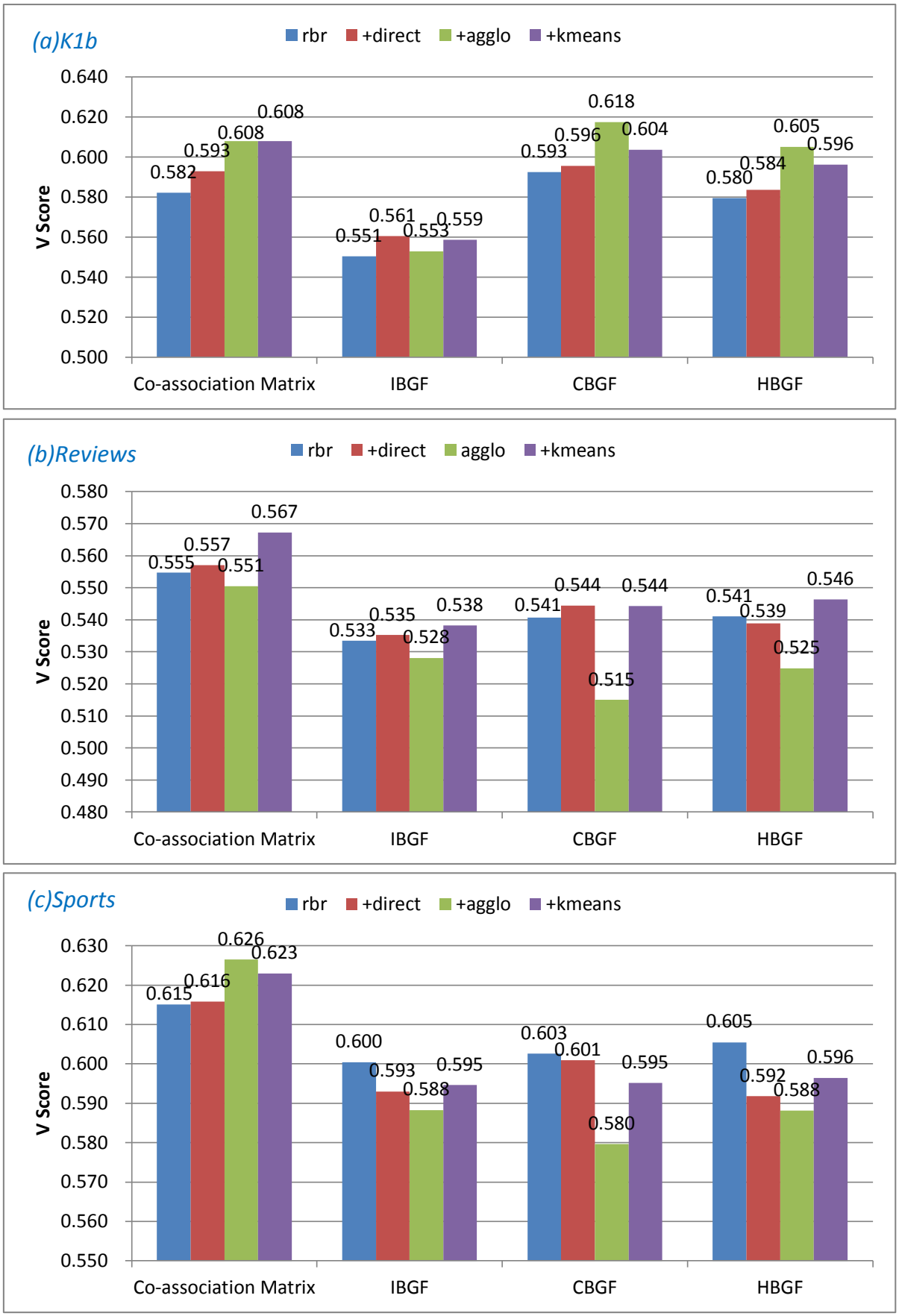


Figure 6.10: Impact of MaCC by incrementally adding clusterers using 4 serial algorithms: repeated bisectional (rbr), direct k-way (direct), agglomerative (agglo) and K-means (kmeans)

Figure 6.11: Impact of MaCC by incrementally adding clusterers ranked by internal measure "silhouette coefficient"

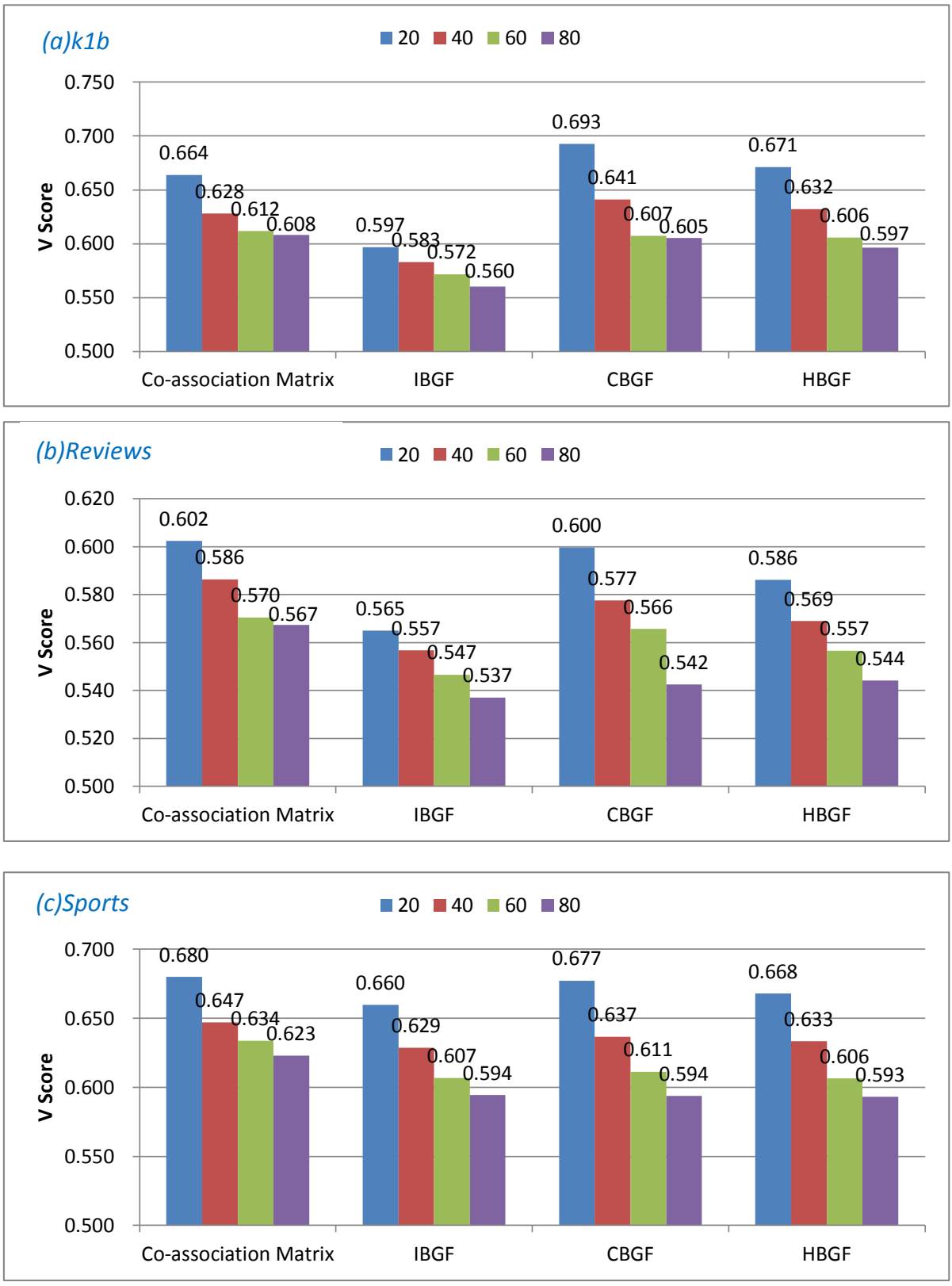


Figure 6.12: Impact of MaCC by incrementally adding clusterers ranked by external measure "V-measure"

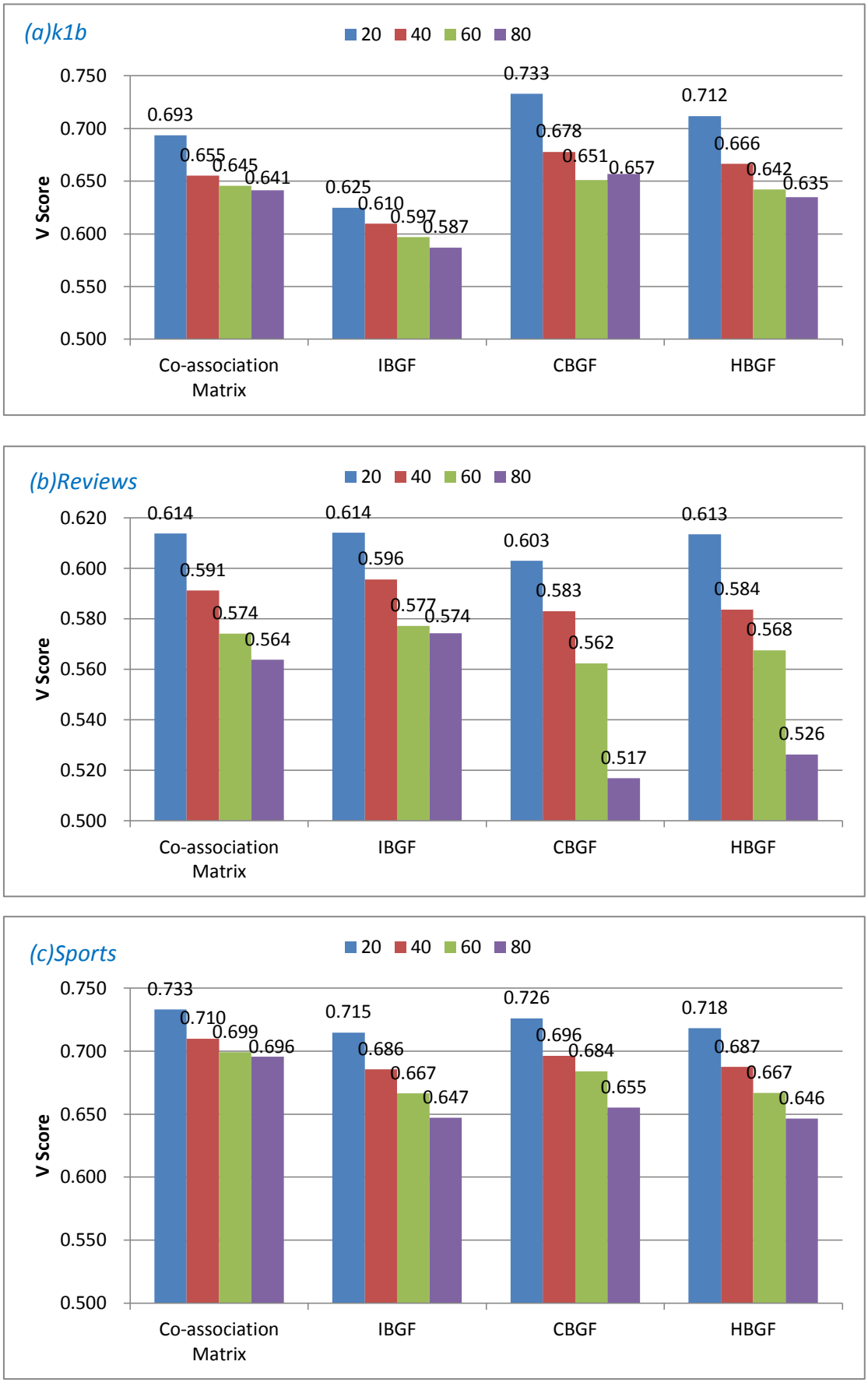


Figure 6.13: Impact of MiMaCC

Chapter 7

Name Entity Clustering: Another Case Study of Collaborative Clustering

Part of the work in this chapter including name disambiguation classification, name variation identification and baseline of name entity entity is based on my 2011 summer internship work at IBM.

7.1 Problem Formulation

Let $\mathcal{Q} = \{q_1, \ldots, q_n\}$ denote the set of n instances. Each instance $q = (q.id, q.string, q.text)$ is a triple consisting of instance id ($q.id$), name string ($q.string$) and context document ($q.text \in \mathcal{C}$) in which $\mathcal{C}$ is source document corpus. The goal of name entity clustering is to generate a hard (non-overlapping) clustering for $\mathcal{Q}$, i.e., $\mathcal{Q} = (Q_1, ..., Q_K)$ such that

(1) $Q_i \neq \emptyset$ for $i \in \{1, ..., k\}$.

(2) $Q_i \cap Q_j = \emptyset$ for $i, j \in \{1, ..., k\}$ and $i \neq j$.

(3) $Q_1 \cup ... \cup Q_k = \mathcal{Q}$

in which each cluster Q_i refers to an entity.

An example of name entity clustering is shown in Figure 1.5 in Chapter 1.

There are two main challenges in name entity clustering:

(1) name variation identification: the same entity can have multiple mention forms. For example, in one instance, the name string is “Michael Jeffrey Jordan” and in the other instance,

the name string is “Michael Jordan”. However, if the two context documents indicate that they refer to the same person, the name entity clustering system should group the two instances into the same cluster.

(2) name disambiguation: the same name can refer to multiple entities. For example, the name strings in both instances are “Michael Jordan”, however, if their context documents indicate that they refer to two different persons, the name entity clustering system should group the two instances into two different clusters.

Before we solve our target problem of name entity clustering, we study the following two simpler problems, i.e., identifying whether two *name strings* are variants or not, identifying whether two *instances* refer to the same entity or not. Both of the two simpler problems can be formulated as classification problems. We believe that successful solutions of these two problems can finally help to set up a high performance name entity clustering system.

7.2 Related Research

The name entity clustering task defined in this thesis is closely related with other research topics such as person name disambiguation[12], web people search [2], cross-document entity coreference resolution. The common thing in these research topics is that they all can be formulated as a clustering problem. However, person name disambiguation and web people search focus on clustering person name entity mentions into unambiguous entities. Cross-document entity coreference resolution studies not only name entities “Mike Jordan”, but also nominal entities such as “professor” and pronominal entities such as “he” and “she”. The clustered results in cross-document entity coreference resolution are often mentioned as coreference chains, for example, a chain can contain the following mentions, “Mike Jordan”, “professor” and “he”. In this thesis, we focus on name entities, and do not explore nominal entities and pronominal entities.

The work in this thesis can be distinguished from previous research in:

(1) unlike person name disambiguation and web people search, we study not only person name entities, but also organization and geo-political entities.

(2) we take both name disambiguation (same name string referring to different entities) and name variation (same entity using different name strings) into considerations, while previous research focuses more on name disambiguation.

7.3 Name Variation Identification

7.3.1 Problem Formulation

We first study name variation identification which can be formulated as a classification problem: given a pair of names, classify them into variant or non-variant. Successful solutions of name variation detection will help instances which may not share the same name forms to be potentially clustered. Without this step, two instances with two different name forms can not be clustered.

7.3.2 Approach

Traditional approaches for name variation identification focus on applying various string similarity measures[22], in this thesis, we intend to apply semantic resources(i.e., Wikipedia) and manually designed rules to identify whether two name strings are variants or not.

We implemented a rule-based classifier consisting of a pipeline of prioritized checkpoints. At each checkpoint, if the two name strings satisfy a given condition, the classifier outputs "variant" or if the pair satisfies a given filtering rule, it outputs "non-variant".

The prioritized checkpoints are defined as follows:

1. Checkpoint 1 is a filtering rule

Filtering 1: two mention forms are both acronyms (a name consisting of all capital letters), and they are not the same in the spelling. For example, "*ABC*" and "*ABT*" can not be variants of each other.

If the filtering rule is satisfied, it outputs non-variant; otherwise, the name pair continues to the next checkpoint.

2. Checkpoint 2 utilizes Wikipedia redirect resource

A redirect page in Wikipedia provides a link to its target entity page for a given alternative name. For example, "*Barack Hussein Obama*" is redirected to "*Barack Obama*".

Let p be the target entity page of a given name if it exists, and $R = \{r_1, r_2, ..., r_N\}$ be the set of redirect pages that are redirected to the target page. We make a string set by assembling the titles of those redirect pages together with the target entity page. For example, for name "Bob Dole", we can obtain such a set as $\{$*Bob Dole,Bob Dole Jokes,Bob dole, Bob doles, Robert Dole, Robert J. Dole, Robert Joseph Dole, Senator Bob Dole*$\}$

If the pair satisfies the following condition, the classifier outputs variant:

Condition 1: both names are in the same set.

3. Checkpoint 3 utilizes Wikipedia disambiguation resource

A disambiguation page in Wikipedia lists all possible links to their target entity pages for an ambiguous name. For example, the disambiguation page of "*ABC*" contains the list of "*American Broadcasting Company*", "*Australian Broadcasting Corporation*", etc.

If a given name has a corresponding disambiguation page, we collect its all possible disambiguated names from the page. Therefore, each name corresponds to a set of disambiguated names (the set could be empty). For example, for name "*Central News Agency*", we can obtain such a set as $\{$*CNA (bookstore),Central News Agency (London), Central News Agency (Republic of China), Korean Central News Agency*$\}$.

If the pair satisfies the following condition, the classifier outputs variant:

Condition 2: one name is in the set of the disambiguated names of the other name.

4. Checkpoint 4 utilizes expanded names extracted from large corpus

If a name is an acronym, we use Lucene search API to retrieve the top 1000 documents (each of which contains the acronym) from the source text corpus prepared for the KBP 2010 evaluation [50]. We extract all unique expanded names that match the following pattern: $expanded\_name(acronym)$. We consider a name as an expanded name if it satisfies the following criteria: (1) the first word in the expanded name starts with capital letter; (2) the length of the expanded name is shorter than the length of acronym multiplying by 20; (3) there are no two continuous words starting with non-capital letter in the expanded name.

Although some of the expanded names found from a large corpus may overlap those ex-

tracted from Wikipedia redirect or disambiguation page, we do find some new names that can not be extracted from Wikipedia. For example, for name "*ABT*", we extracted a new name variant from the corpus: *Agricultural Bank of Taiwan*.

If the pair of names satisfies the following condition, the classifier outputs variant:

Condition 3: one name is in the set of the expanded names of the other name.

5. Checkpoint 5 utilizes coreference names extracted from large corpus

For a given name, we apply the IBM KLUE toolkit [43] to obtain coreference names from the top 300 documents that contain the target name. We only pick the coreference names with the type of "NAM" (proper names). For example, for name "*Air Macau*", we obtained three new name variants through within-document coreference resolution: *Air Macau* , *China Airlines*, *CAL*, *Macau*. However, due to the fact that within-document coreference resolution system may introduce errors, the extracted name variants may not be correct, as shown in the above example, "*Macau*" could not be a good name variant of "*Air Macau*".

If the pair of names satisfies the following condition, the classifier outputs variant:

Condition 4: one name is in the set of the coreference names of the other name.

6. Checkpoint 6 contains several specific conditions

(1) ***Condition 5***: one name is an acronym, and the other is not. Meanwhile, we can get the spelling of the acronym by connecting the first capital letter of each word in the non-acronym name. For example, one name is "*IBM*", the other name is "*Internal Business Machine*".

(2) ***Condition 6***: one name is an acronym, and the other is not. Meanwhile, the spelling of the non-acronym name starts with the acronym. For example, "*RIA*" and "*RIA Novosti*".

(3) ***Condition 7***: the two names share at least 3 same words, for example, "*Tianjin FAW Toyota Engine*" and "*Tianjin FAW Toyota Motor Co. Ltd.*"

(4) ***Condition 8***: both names are person names, and we can find at least one overlapping spelling from the two sets of coreference names.

If any of the above conditions are satisfied, the classifier outputs variant.

7. The final checkpoint consists of two lenient conditions.

Condition 9: the Levenshtein distance between the two names is smaller than some threshold ($Threshold = 2$).

Condition 10: one name string is a substring of the other.

If none of the above conditions are satisfied, the classifier outputs non-variant.

7.3.3 Dataset and Measure

We created a data set containing pairs of 514 unique name strings extracted from KBP 2009 entity linking evaluation corpus and the goal of name variation identification is to classify each pair into variant or non-variant. A name string, together with a associated context document, make up a query in the 2009 evaluation corpus. There are 3904 queries in the KBP 2009 evaluation corpus and each query is either linked to an entry in the knowledge base (KB) or assigned a NIL id. In order to tell whether two name strings are variants or non-variants automatically, we wrote a program that checks through all the queries each of which contains one of the two name strings. If there are two queries linked to the same KB id or the same NIL id and the contained two name strings are different, then we say the two name strings are variants. If there are not any two queries which can be linked to the same id, then the two name strings are non-variants. As a result, we created 406 "variant" pairs, and 131435 "non-variant" pairs. It is worth noting that in those 131435 non-variant pairs, some pairs are actually variants, however they can not be detected by the program, simply because we only check the cases contained in those 3904 queries. To create more accurate answer keys without making huge efforts in manually checking the majority of "non-variant" pairs, we first applied the full functional rule-based classifier described above to produce the initial outputs and then manually reviewed "variant" pairs detected by the classifier which were annotated as "non-variant" in answer keys. We found that in most cases, we can remedy the errors that are introduced by the automatic program from which the answer keys were created. Furthermore, since the number of "variant" pairs produced by the classifier is not overwhelming, we can quickly check those pairs in short time.

We use precision, recall, and F-measure to evaluate the performance of the rule-based classifier.

The confusion matrix is shown in table 7.1.

Precision is computed as $P = \frac{a}{a+b}$.

gold / *system*	*variant*	*non-variant*
variant	a	b
non-variant	c	d

Table 7.1: Confusion matrix of name variant classifier.

Recall is computed as $R = \frac{a}{a+c}$.

F-measure is computed as $F = \frac{2*P*R}{P+R}$.

7.3.4 Experimental Results

The experimental results are showed in Table 7.2 and Figure 7.1. There are two major columns at each major step by incrementally adding more checking points, one of which shows *P* (Precison), *R* (Recall) and *F* (F-measure) before manual reviewing of key answers (i.e., using automatically generated answer keys), the other shows the performance after manual reviewing.

	before manual review			after manual review		
	P	R	F	P	R	F
baseline	0.35	0.32	0.33	0.42	0.28	0.34
(+acronym filtering)	0.40	0.31	0.35	0.48	0.28	0.35
(+redirect)	0.48	0.48	0.48	0.57	0.43	0.49
(+disambiguation)	0.47	0.55	0.51	0.63	0.57	0.60
(+expanded)	0.48	0.59	0.53	0.65	0.60	0.63
(+coref)	0.47	0.65	0.54	0.63	0.66	0.65
(+specific rules)	0.53	0.71	0.61	0.79	0.79	0.79

Table 7.2: Impact of various resources and rules in the task of name variation detection

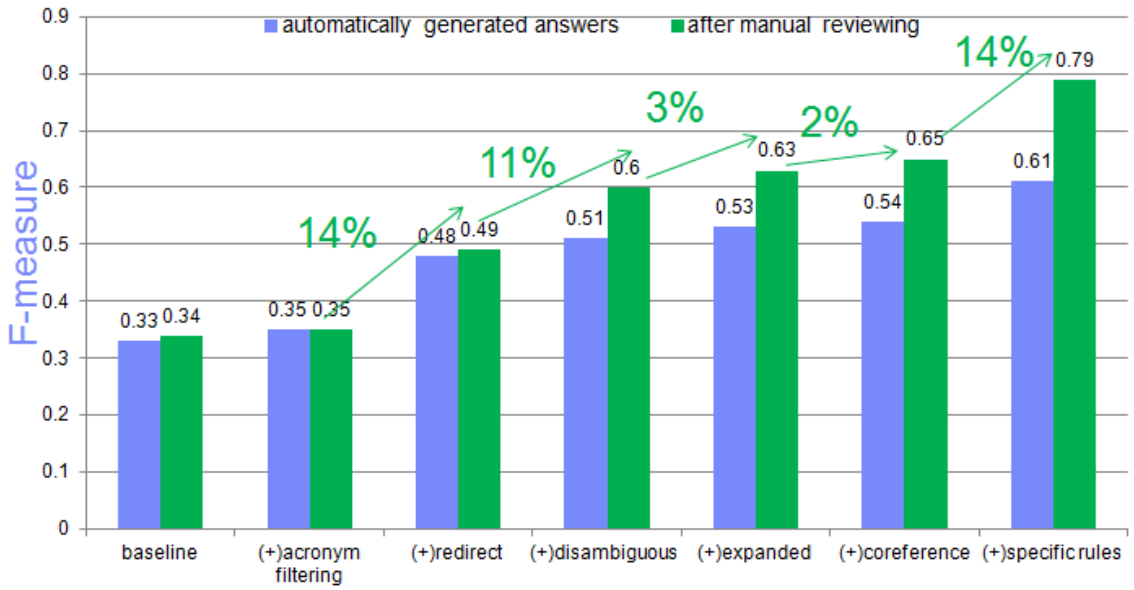


Figure 7.1: Impact of various resources and rules in the task of name variation detection

The results show that redirect and disambiguation resources are indeed useful to identify name variants (14% and 11% absolute gains after manual cleaning). The expanded names and coreference names also improve the performance which clearly shows that Wikipedia resources are not complete. Specific rules (condition 7,8,9,10) also help the performance by 13% gains.

The errors from the rule-based classifier come from two-fold sources:

Type I: two name strings are in fact "variants", however, the classifier outputs "non-variants";

Type II: two name strings are in fact "non-variants", however, the classifier outputs "variants".

There are several reasons that lead to the first type of errors:

(1) the Wikipedia resources can not cover all the cases, furthermore, the pattern matching program that detects expanded names for acronyms and the program for within-document coreference resolution are far from perfect. Therefore, quite some of the name variants are not identified. We further split the cases into 3 categories by entity types, namely, person, organization and GPE. Some **person** related variants are not identified because of the shortage of resources for detecting nicknames or birthnames. For example, "*Angela Dorothea Kasner*" is the birthname of "*Angela Dorothea Merkel*" who can also be mentioned as "*Angie Merkel*" or "*Maggie Merkel*" or "*The Iron Frau*". **Organization** related variants are hard to detect if such resources as stock symbols are not collected and applied. For example, "*Chunghwa Telecom Co., Ltd.*" or its shorter form "*Chunghwa Telecom*" is listed using stock symbol "*CHT*" at the New York Stock Exchange (NYSE). Furthermore, the rule in *Condition 5* which works for most of the time can not identify the variants sometimes. For example, concatenating the capital letters in "*the Islamic Conference Organization*" can not obtain its variant of "*OIC*" (the Organization of Islamic Conference). Other examples include "*SAC*" and "*Southern Ammunition Co. Inc.*", "*NSC*" and "*United States National Security Council*". Some organizations also have nicknames, for example, "*Galatasaray SK*" is called "*The Lions*". **Geopolitical** related variants are hard to identify because some nicknames are only known in local people, or exist for some period in the history. For example, "*Nablus*", a Palestinian city, was called "*Little Damascus*" since around the 10th-century.

(2) The filtering rule in *Filtering 1* works in most of the time, however, it may fail some-

times. For example, "*TSE*" may mean Taiwan Stock Exchange, and "*TSEC*" which is the acronym of "Taiwan Stock Exchange Corporation", can also mean Taiwan Stock Exchange.

The error distribution by categories is presented in Figure 7.2 (a). To overcome these errors in the future, we need to collect more resources for persons, organizations and GPEs respectively.

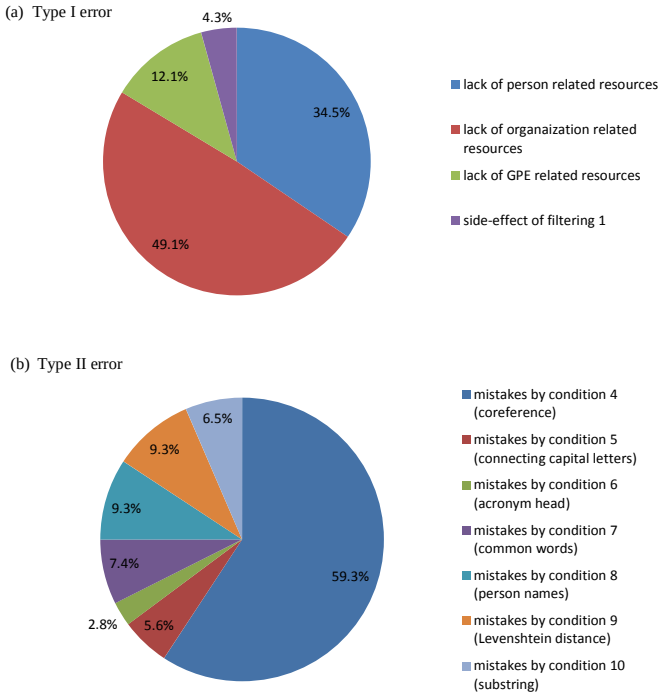


Figure 7.2: Error distribution by categories (Type I error: missing variants, Type II error: wrong variants)

There are also several reasons that lead to the second type of errors:

(1) due to the imperfect performance of within-document entity coreference resolution system, satisfying *condition 4* leads to errors. For example, the resolver considers the pair "Macao" and "Air Macao" as name variants which are wrong.

(2) satisfying *condition 5* may also introduce some errors, for example, "CCP" and "CNS Confectionery Products", "CC" and "Chelsea Clinton" which is a person name.

(3) satisfying *condition 6* leads to errors , for example, "CPN" and "CPN(UML)"

(4) satisfying *condition 7* leads to errors, for example, "Communist Party of India (Marxist-Leninist)" and "Communist Party of Nepal (Marxist-Leninist)" which shares 4 common words, but they are not name variants.

(5) satisfying *condition 8* leads to errors, for example, "Mohammad Daud Khan" and "Mohammad Khan Junejo".

(6) satisfying *condition 9* leads to errors, for example, "MDC" and "MoD" are not variants, but their Levenshtein distance is 2.

(7) satisfying *condition 10* also leads to errors, for example, "John Dewey" and "John Dewey High School" in which the former is a substring of the latter.

The error distribution by categories is presented in Figure 7.2 (b). To overcome these errors in the future, we need to design more complicated rules, however, the dilemma is that the gain in precision (reducing type II error) is usually at the cost of the loss in recall (increasing type I error).

7.4 Name Disambiguation Detection

7.4.1 Problem Formulation

Given a pair of instances in which they share the same name string but are associated with two different context documents, classify them into *coreference* or *non-coreference*. Successful solutions of name disambiguation classification will help construct better similarity functions since a supervised classifier can normally provide a probability that tells to what extent the two instances refer to the same entity.

Since the two instances are associated with documents which provide lots of information to help disambiguate, therefore, we first preprocess those context documents to extract some important information. This preprocess step was also presented in [12] and some information extracted by ours overlaps with theirs; however, we go beyond their work by not only studying how the direct extracted information can help disambiguate persons, but also studying how the more complicated features derived from the extracted information can help disambiguate persons, organizations and GPEs.

7.4.2 Preprocessing Step

For each context document, we extract the following information using the IBM KLUE toolkit [43].

1. target mentions

Target mentions are a set of entity mentions one of which has the form of the name string in the instance, and the others are the coreference mentions. For example, Target mentions often contain some useful information about the ambiguous object, for example, the title of a person. Therefore, it is possible to disambiguate names by leveraging the information in the conflicting nominal target mentions (e.g., professor "Barack Obama" in one document and president "Barack Obama" in the other document).

Once we find the target mentions, we can locate the local sentences in which target mentions occur.

Sometimes, we can not find any such target mentions using the IBM toolkit because it can not detect any entity mentions that has the name string. In the data set we used, we observed that in about 7% cases, we do not find any target mentions. In such cases, we use exact string matching to locate the local sentences in which the name string occurs.

2. local entity mentions

Local entity mentions are those mentions (excluding target mentions) that occur in the local sentences that contain at least one of the target mentions. It is worth noting that we only extract name ("NAM") and nominal ("NOM") entity mentions. We can also take coreference information of those entity mentions into considerations, i.e., from the entity of each local mention, we get its canonical form[1].

Local entity mentions often contain useful information that is related with the ambiguous name, such as related persons, locations.

3. local noun phrases

We also extract the noun phrases that occur in the local sentences that contain one of the target mentions. Though there could be some overlaps between local noun phrases and local entity mentions, local noun phrases may contain extra information that does not exist in local entity mentions.

4. non-local (global) entity mentions

Global entity mentions are those mentions that occur in the whole document excluding target mentions and local entity mentions. Non-local entity mentions often provide implicit

[1] canonical form is the longest "NAM" mention

and background information for the ambiguous object. As we did for local entity mentions, we only get the canonical form of each entity.

5. person/organization/GPE entity mentions We extract person, organization and GPE entity mentions from the whole document. We only get the canonical form of each entity.

6. relation mentions

We extract relation mentions in which the target mentions are involved. We collect relation types from those relation mentions, meanwhile we also collect the participant entity mentions that are involved in the same relation mentions with those target mentions.

Using the same example presented in [12], we can extract the information as shown in Table 7.3.

<table>
<tr><td colspan="3">Name: John Smith
Document:
S1 HOPE MILLS — Commissioner Tonzie Collins has been banned from a town restaurant after an alleged run-in with two workers there Feb. 21. ……
S4 “In all fairness, that is not a representation of the town,” said Jenny Thomas, manager at Andy’s Cheesesteaks in the Village Shopping Center on Hope Mills Road. ……
S16 Hope Mills police Capt. John Smith said based on what he read in the statements, no criminal violation was committed.
S17 “Based on what the individuals involved said, there was no direct threat,” Smith said.
S18 He and Thomas said they don’t think Collins intentionally left without paying his bill. ……</td></tr>
<tr><td colspan="2">target mentions</td><td>John Smith S16, police Capt. S16, Smith S17, ~~he~~ S18</td></tr>
<tr><td rowspan="2">local mentions</td><td>mentions in local sentences</td><td>Hope Mills S16, Thomas S18, Collins S18</td></tr>
<tr><td>coreference mentions in canonical forms</td><td>Hope Mills S16, Jenny Thomas S4, Tonzie Collins S1</td></tr>
<tr><td colspan="2">local noun phrases</td><td>Hope Mills S16, police Capt. S16, statements S16, criminal violation S16, individuals S17, threat S17, bill S18</td></tr>
<tr><td colspan="2">global mentions</td><td>Commissioner S1, restaurant S1, manager S4, Andy’s Cheesesteaks S4, Village Shopping Center S4, Hope Mills Road S4</td></tr>
<tr><td colspan="2">person mentions</td><td>John Smith S16,Jenny Thomas S4 ,Tonzie Collins S1</td></tr>
<tr><td colspan="2">organization mentions</td><td>restaurant S1 ,Andy’s Cheesesteaks S4</td></tr>
<tr><td colspan="2">GPE mentions</td><td>Village Shopping Center S4,Hope Mills Road S4,Hope Mills S16</td></tr>
<tr><td colspan="2">relations</td><td>[located at$^{\text{relation type}}$: Hope Mills$^{\text{participant argument}}$]</td></tr>
</table>

Note: “he” in sentence 18 is a coreference of “John Smith”, although it is excluded from the set of target mentions, we still extract local entity mentions from S18.

Table 7.3: An example of preprocessing a document associated with an instance

7.4.3 Features

Based on the above extracted information, we designed the following features as tabulated in Table 7.4. Basically, for each document, we can obtain a series of string vectors based on the extracted information. Most of features are computed by comparing corresponding vectors from two documents. For example, if there are conflict "NOM" target mentions between two vectors from two documents, then the two instances are likely to refer to the same entity. This comes from the intuition that "professor" Barack Obama should be distinguished from "president" Barack Obama. For the other example, if there are more overlapping local mentions or local noun phrases, the two instances are more likely to refer to the same entity. Some features take integer values, some features take real values in which cases they will be quantized into bins. When comparing the common items in two vectors, we also apply two strategies, one is based on exact matching, the other is based on exact matching by applying softTFIDF [23] (similarity threshold was set to be 0.65, i.e., if the ScoreTFIDF score for two name strings is higher than 0.65, we consider them as matched).

• Features of np, tc, all, l1, l2, p, o, g, r, arg are computed as the number of exact matched items between two vectors.

• Features of nps, alls, l1s, l2s, ps, os, gs, args are computed using softTFIDF [23].

• Features with names ended by "j" are first computed by Jaccard formula: $\frac{|A \cap B|}{|A \cup B|}$ where A and B are two vectors, and then quantized into bins. We add "1" after the original feature name (e.g., "npsj1") to represent 10-bin quantization (scale the real value by 10 times), and "2" after the original feature name to represent 20-bin quantization.

7.4.4 Classification Model

The classification model we applied is maximum entropy model which can be expressed as

$$P(L|q_i, q_j) = \frac{\exp(\sum_{k=1}^{n} \lambda_k f_k(q_i, q_j, L))}{Z(q_i, q_j)}$$

where $f_k(\bullet, \bullet, L)$ is a feature and λ_k is its weight; $Z(\bullet, \bullet)$ is a normalized factor. The output of L has two possible values, 1 for coreference which means that the two queries refer to the

Features	Description
np	the overlap number of noun phrases in the two noun phrase vectors
tc	the conflicting number of target mentions in the two target mention vectors
all	the overlap number of mentions in the two non-local mention vectors
l1	the overlap number of mentions in the two local entity mention vectors (without coreference)
l2	the overlap number of mentions in the two local entity mention vectors (with coreference)
p	the overlap number of mentions in the two person entity mention vectors
o	the overlap number of mentions in the two organization entity mention vectors
g	the overlap number of mentions in the two GPE entity mention vectors
r	the overlap number of relation types in the two relation type vectors
arg	the overlap number of relation arguments in the two argument vectors
npj	jaccard similarity between the two noun phrase vectors
allj	jaccard similarity between the two non-local entity mention vectors
l1j	jaccard similarity between the two local entity mention vectors (without coreference)
l2j	jaccard similarity between the two local entity mention vectors (with coreference)
pj	jaccard similarity between the two person entity mention vectors
oj	jaccard similarity between the two organization entity mention vectors
gj	jaccard similarity between the two GPE entity mention vectors
argj	jaccard similarity between the two argument vectors
nps	the overlap number of noun phrases (SoftTFIDF) in the two noun phrase vectors
alls	the overlap number of mentions (SoftTFIDF) in the two non-local entity mention vectors
l1s	the overlap number of mentions (SoftTFIDF) in the two local entity mention vectors (without coreference)
l2s	the overlap number of mentions (SoftTFIDF) in the two local entity mention vectors(with coreference)
ps	the overlap number of mentions (SoftTFIDF) in the two person entity mention vectors
os	the overlap number of mentions (SoftTFIDF) in the two organization entity mention vectors
gs	the overlap number of mentions (SoftTFIDF) in the two GPE entity mention vectors
args	the overlap number of mentions (SoftTFIDF) in the two argument vectors
npsj	jaccard similarity (SoftTFIDF) in the two noun phrase vectors
allsj	jaccard similarity (SoftTFIDF) in the two non-local entity mention vectors
l1sj	jaccard similarity (SoftTFIDF) in the two local entity mention vectors (without coreference)
l2sj	jaccard similarity (SoftTFIDF) in the two local entity mention vectors (with coreference)
psj	jaccard similarity (SoftTFIDF) in the two person entity mention vectors
osj	jaccard similarity (SoftTFIDF) in the two organization entity mention vectors
gsj	jaccard similarity (SoftTFIDF) in the two GPE entity mention vectors
argsj	jaccard similarity (SoftTFIDF) in the two argument vectors
tfidf	cosine similarity between the two documents
esim	entity similarity between the two documents

Table 7.4: Features of maximum entropy model for name disambiguation

same entity, 0 for not coreference.

We utilized the implementation of maximum entropy model in OpenNLP toolkit[1].

7.4.5 Dataset and Measure

Similar with name variation experiments, our data sets for name disambiguation are also created based on KBP 2009 entity linking evaluation corpus. In the 2009 corpus, each query (consisting

[1] http://incubator.apache.org/opennlp/

of a name and a context document) is either linked to an entry in the knowledge base (KB) or assigned a NIL id. There are 514 names distributed among 3904 queries including 96 person names, 358 organization names, 78 GPE names[1]. Some names are ambiguous, which means the queries with those names can be clustered into 2 or more clusters, while some other names are unambiguous which means all the queries with those names are from the single cluster.

We constructed two datasets by forming pairs of queries. The first dataset covers 101 ambiguous names in those 514 names including 27 persons, 73 organizations, and 19 GPEs. The second dataset is constructed by using all the 514 names. The answer keys are created based on the following rule: if the two queries in the pair are linked to the same KB id or assigned by the same NIL id, then they are coreference, otherwise non-coreference. Table 7.5 shows the number of coreference and non-coreference pairs for dataset 1 and dataset 2 respectively. It shows that GPE names are much less ambiguous than PER names or ORG names as we observed that a majority of the pairs belong to the class of "coreference". It is also worth noting that the number of ORG pairs dominate PER and GPE pairs.

		All	PER	ORG	GPE
dataset 1	#coreference	10453	790	8909	754
	#non-coreference	14404	2054	12171	179
dataset 2	#coreference	21404	2132	15330	3942
	#non-coreference	14404	2054	12171	179

Table 7.5: Class distribution of dataset1 and dataset2.

We use Precision (P), Recall (R), and F-measure to evaluate the performance of supervised classifier.

The confusion matrix is shown in table 7.6.

gold / ***system***	***coreference***	***non-coreference***
coreference	a	b
***non*-coreference**	c	d

Table 7.6: Confusion matrix of name disambiguation classifier.

Precision is computed as $P = \frac{a}{a+b}$.

Recall is computed as $R = \frac{a}{a+c}$.

F-measure is computed as $F = \frac{2*P*R}{P+R}$.

[1] some names have multiple entity types

7.4.6 Experimental Results

7.4.6.1 Feature Contribution

The goal of our first set of experiments is to study the feature contributions in a maximum entropy model. We actually implemented four maximum entropy models, one single maximum entropy model which does not distinguishing entity types, and three individual maximum entropy models for three entity types *PER*, *ORG* and *GPE* respectively. We applied information gain[1] to measure the contribution of each feature. The higher information gain a feature has, the greater contribution (more importance) the feature has in the model. Information gain can be formally defined as follows:

Let T be a set of training set, each instance in the set can be expressed as $(x, y) = (x_1, x_2, ..., x_k, y)$, where $x_i = a \in Vals(x_i)$ indicates the i_{th} feature has value a and the set of values that x_i can take is $Vals(x_i)$, and y is the corresponding class label. The information gain for each feature x_i can be computed as

$$IG(T, x_i) = H(T) - \sum_{a \in Vals(x_i)} \frac{|\{x \in T | x_i = a\}|}{|T|} H(\{x \in T | x_i = a\}) \tag{7.1}$$

where $H()$ is defined as the entropy of the set.

We utilized the implementation of information gain included in the Weka toolkit[42]. Table 7.7 shows the top 30 features ranked by information gain from high to low.

Table 7.7 shows that for *ORG* and *GPE* names, GPE features (feature names starting with "g") are more important than the others, meanwhile, global features (feature names starting with "all") have more contributions than local features (feature names starting with "l"). For *PER* names, local features dominate the global features and PER features (feature names starting with "p") are more important.

By analyzing the data, we observed that

(1) if the name in the pair of queries is an *ORG* and *GPE* name, and the two associated documents share some common GPE related mentions, then the pair is likely to be coreference. For example, "Newark" is an ambiguous city name, however, if in the two documents, it is

[1] an alternative synonym of Kullback-Leibler divergence.

All	PER	ORG	GPE
gj2	tfidf2	gj2	tfidf2
gsj2	tfidf1	gsj2	gj2
gsj1	l2sj2	gsj1	gj1
gj1	l2s	gj1	allj2
allj2	l1s	allj2	gsj1
allsj2	l1sj1	allsj2	gsj2
tcj2	l2sj1	tcj2	l2s
g	l1sj2	g	l2sj2
tcj1	ps	tcj1	l2sj1
allsj1	psj2	allsj1	allj1
allj1	p	all	l2j1
all	l1	allj1	l2j2
gs	np	gs	l2
tfidf2	pj2	tfidf2	l1s
tfidf1	psj1	tfidf1	g
alls	l2	alls	l1j2
esim2	nps	esim2	l1j1
l2sj2	l1j2	esim1	l1sj1
l2sj1	l2j2	l2sj2	l1sj2
esim1	allj2	l2sj1	l1
l2s	tcj2	l2j2	tfidf1
l2j2	allsj2	l2j1	osj1
l2	tcj1	l2	args
l2j1	allsj1	os	argsj1
l1sj2	l1j1	l2s	argsj2
l1j2	pj1	pj2	np
l1sj1	l2j1	p	osj2
l1	allj1	psj2	all
l1j1	all	l1j2	gs
l1s	gsj2	psj1	oj1

Table 7.7: Feature contribution ranked by information gain from high to low.

followed by the same state name, “Delaware” or “DE”, then it is likely that the two documents talk about the same entity. It is also common that when a document mentions an organization, it is also likely to mention its location.

(2) if the name in the pair of queries is a *PER* name, and the two associated documents mention common person names, then the pair is likely to be coreference. It is true that a person together with others involved in some event are often reported in a document.

All the above observations explain why the features have different contributions. However,

Table 7.7 also shows that relation related features (e.g., "r", "arg") which we expect to play important roles in disambiguating names did not rank high or even appear in our list. The reasons may be that: (1) relation extraction system is not good enough so that it introduces noises, (2) those features suffer more from the data sparsity of extracted feature values.

Table 7.7 shows that the top most important features for one single model ("All" column) are quite similar with those for *ORG* model. The reason is that the number of *ORG* names dominate the other two types in our dataset which indicates that it is necessary to develop a separated maximum entropy model for each entity type.

	single model			3 models			3 models with reduced features			3 models with reduced features + acronym handling		
	P	R	F	P	R	F	P	R	F	P	R	F
dataset 1	0.632	0.760	0.682	0.680	0.815	0.735	0.680	0.823	0.739	0.690	0.820	**0.742**
dataset 2	0.752	0.877	0.801	0.785	0.899	0.832	0.795	0.892	0.836	0.799	0.903	**0.843**

Table 7.8: Performance of supervised classifier.

7.4.6.2 Separate Models Vs. Single Model

In order to validate that the separated maximum entropy models for three entity types can work better than a single maximum entropy model, we conducted five-fold cross validation on the two datasets discussed above. The experimental results in Table 7.8 show that the separate models can significantly work better than single model. For dataset 1, the performance gains is 5.3% in F-measure and it is statistical significance at a 99.9% confidence level by conducting Wilcoxon Matched-Pairs Signed-Ranks test. For dataset 2, the performance gains is 3.1% with statistical significance at a 99.9% confidence level.

7.4.6.3 Impact of Separate Models with Reduced Features

We also experimented 3 separate maximum entropy models with the top 30 selected features for each entity type. Table 7.8 shows it obtained slight performance gains of 0.4% for dataset 1 and 0.4% for dataset 2. However, both gains are statistical significant at a 99.9% confidence level by conducting Wilcoxon Matched-Pairs Signed-Ranks test.

7.4.6.4 Impact of Handling Acronyms

We noticed that there are quite a few acronyms for organization names. Therefore, we designed a rule based classifier and put it ahead of the supervised maximum entropy model to process the pairs with acronym names. For each acronym, we first searched possible expanded names in the two associated documents. The rule based classifier works as follows:

(1) if expanded names can be searched in the two documents and they are exactly the same, the classifier outputs coreference.

(2) if expanded names can be searched in the two documents and they are not the same, the classifier outputs non-coreference.

Table 7.8 shows that this extra processing can obtain a slight performance gains. The gains are statistical significance at a 99.9% confidence levels for both dataset 1 and dataset 2.

7.5 Name Entity Clustering

7.5.1 Two Baseline systems

Baseline 1. (all-in-one) For each name, we cluster all the instances into one single cluster.

Baseline 2. (one-in-one) For each name, we cluster each instance into a cluster.

7.5.2 Clustering Algorithms and Similarity Functions

Similar with the approaches applied in document clustering discussed in Chapter 6, we applied 21 baseline clustering algorithms, including 9 agglomerative clustering algorithms (3 based on linkages and the other 6 each of which optimizes a specific internal measure), 12 partitional clustering algorithms (6 repeated bisectional clustering algorithms each of which optimizes a specific internal measure and 6 k-way direct clustering algorithms each of which also optimizes a specific internal measure). The symbols to represent the 9 agglomerative clustering algorithms are $slink$,$clink$,$alink$,I_1, I_2, ε_1, H_1, H_2 respectively, the symbols to represent the 6 repeated bisectional clustering algorithms are rI_1, rI_2, $r\varepsilon_1$, rH_1, rH_2 and the symbols to represent the 6 k-way direct clustering algorithms are dI_1, dI_2, $d\varepsilon_1$, dH_1, dH_2.

We studied two settings of each clustering algorithm, one is to use fixed K (assuming that we know the prior of the number of clusters), the other is to compute Silhouette Coefficient to automatically determine the number of clusters. The algorithm for determining the number of clusters by applying an internal measure has been presented in Algorithm 5 in Chapter 5. It is worth noting that in document cluster, we only cluster the documents into fixed number of clusters by assuming that the prior K is known.

Besides the cosine similarity function and correlation similarity applied in document clustering, we include another two supervised approaches to compute the coreference likelihood of two instances, one is based on maximum entropy model as discussed in section 7.4.4, and the other is based on Support Vector Machine for which we utilized the LibSVM[1] toolkit and applied default parameters (e.g., linear kernel function). We use symbols *cos*, *cor*, *maxen*, *svm* to represent the four similarity functions.

The features applied in the above two models have been presented in Table 7.4. It is worth noting that maximum entropy model we utilized can only accept categorical numbers (so the real values are quantized), while SVM can accept real values.

In total, we can produce $84 = 21 \times 4$ clustering results by taking the combinations of 21 clustering algorithms and 4 similarity functions.

7.5.3 Experiments

7.5.4 Dataset and Evaluation Metric

We created a dataset based on the following procedure:

(1) we first collected all the queries from KBP 2009 evaluation corpus, 2010 entity linking evaluation corpus and 2011 entity linking training corpus. We obtained 6652 queries (i.e., instances) distributed in 1379 names.

(2) we wrote a program to automatically pick out names that satisfy the following conditions: (a) the name should be ambiguous, which means the queries that contain the name should be able to be clustered into 2 or more clusters according to the answer keys; (b) the number

[1] http://www.csie.ntu.edu.tw/ cjlin/libsvm/

of queries that contain the name should be larger than 4; (c) the entity types in those queries that contain the name should be consistent and those name with mixed entity types are filtered; (d) besides the context documents in those queries which share a name, we can find at least 5 relevant documents that also contain the name.

As a result, we obtained a dataset consisting of 1686 queries distributed in **106 names**, including **21 person names**, **67 organization names**, and **18 GPE names**.

In some sense, we have simplified the name entity clustering problems from the following aspects:

(1) we assume that the queries which share the same name can be clustered into at least 2 clusters. The reason of filtering unambiguous names is that most of advanced clustering algorithms aim to cluster instances into $K > 1$ clusters. In reality, we also need to handle the case that all the queries should be grouped into one cluster. To solve this problem in the future, we can apply a two-step processing, first we use a binary classifier to determine whether all the queries should be clustered together or they should be further clustered into more than 2 clusters. In the later case, we just apply the clustering algorithms as studied in this thesis.

(2) we assume that the queries which share the same name have a consistent entity type. However, in reality, two queries with the same name can have different entity types, for example, "Chicago" may mean "the City of Chicago" which has a GPE type, but it can mean "Chicago Bulls" which is a football team and has a ORG type. To solve this problem in the future, we can simply first group the queries which share the same name by entity types, because it is obvious that queries with different entity types are unlikely to be clustered together.

We ignored a name if there are fewer and equal to 4 queries that share the name, simply because almost all the baseline clustering algorithms can equally work well on this small number of queries as we have experimented. We also omitted a name if we cannot find at least 5 collaborative documents that contain the name. The motivation of doing so is to experiment the algorithms of micro (instance level) collaborative clustering. If there are too few collaborative instances, the MiCC algorithm may not work well.

By analyzing the dataset, we observed two types of long tail effect:

Type I: As shown in Figure 7.3, most ambiguous names only refer to a few entities (i.e.,

clusters). For example, 39 names (36.8% of the total) only have two clusters, and 91 names (85.8% of the total) have fewer than and equal to 6 clusters. There are only a few names that have extremely large number of clusters.

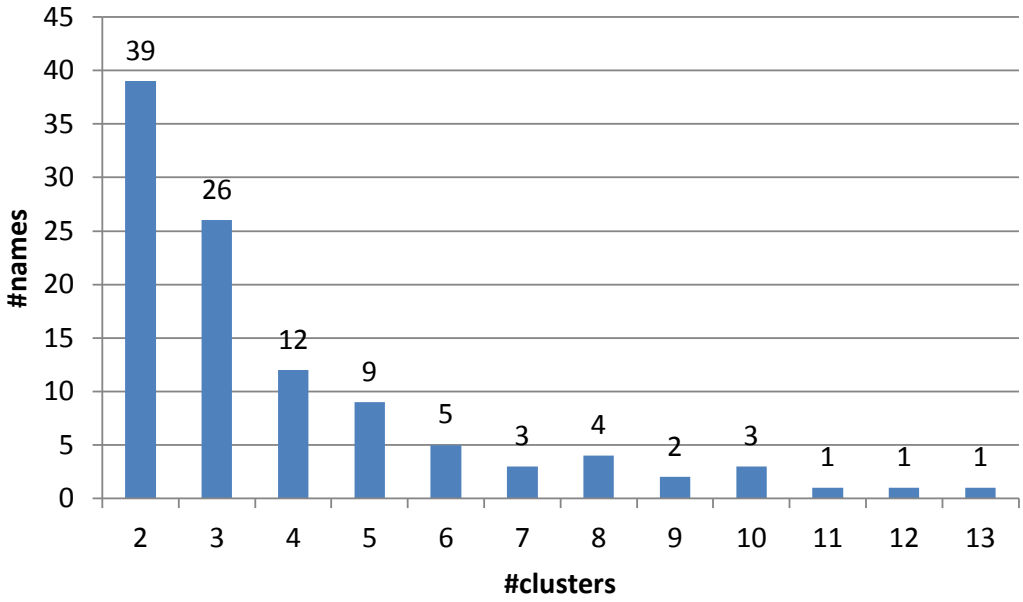


Figure 7.3: Type I long tail effect

Type II: As shown in Figure 7.4, most names have very unbalanced class distribution, in other words, most queries (instances) are clustered in a few large clusters, while the other queries are clustered into small clusters including many singleton clusters. The number on each bar is computed as follows: we rank the clusters for each name from high to low by computing the number of queries contained in each cluster, so the first cluster (as shown in the most left bar in Figure 7.4) always contains the largest number of queries; we then compute the average number of queries for the i_{th} cluster among all the names. The figure shows that the top cluster contains 9.4 queries in average, and the second largest cluster only contains 3.3 queries, and more clusters contains even fewer queries.

We can also compute the coefficient of variation (CV) to measure the skewness of class distribution for each name. By taking all the 106 names into consideration, the maximum CV we obtained is 1.862, the minimum CV is 0 and the average CV is 0.849, standard deviation is 0.411. About 80% of the 106 names have the CV value above 0.5 which indicates that many names have a high unbalanced class distribution. This is intuitive because some entities (e.g., famous persons) are so popular such that the documents mentioning those popular entities dominate the others.

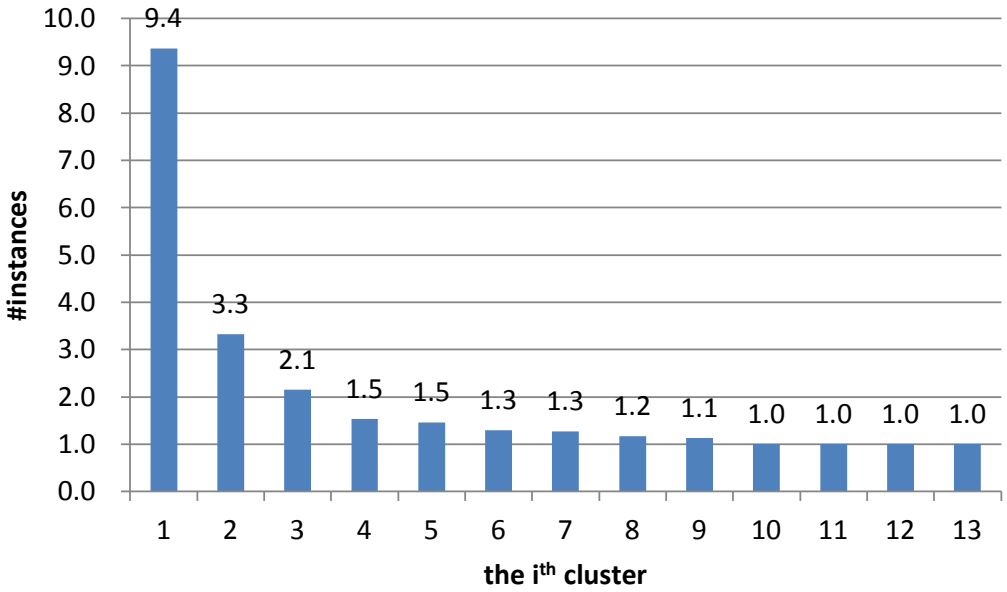


Figure 7.4: Type II long tail effect

As discussed in Chapter 6, the external measure V-measure can capture the "uniform effect", therefore, we applied V-measure to measure the clustering results in our experiments for name entity clustering.

7.5.4.1 Impact of Clustering Algorithms

name	cluster distribution	V-score of one-in-one
George Kennedy	8,1,1,1	0.539
George Young	10,1	0.225
Iron Lady	16,8,4,3,3,2,1,1,1,1	0.662
Li Jie	20,10,3,2,2,1,1,1,1,1	0.604
Michael Kennedy	6,3,2,2,1,1,1	0.766
Mike Kennedy	2,1,1,1,1	0.931
Mohamed Ahmed	2,1,1,1,1,1	0.946

Table 7.9: Examples of performance of one-in-one system

We first tested the performance of two baseline systems, one-in-one and all-in-one. For one-in-one system, each of the instances of each name forms a singleton cluster. Since we measure the performance by V-measure, the value of *homogeneity* in the V-measure is 1 because each cluster contains only members of a single class, however, the value of *completeness* is low because the members in a single class are assigned to different clusters. As discussed in the previous section, the cluster distribution in our dataset follows a long-tail effect, in other words, for many names, the instances are clustered into a few large clusters together with even more

similarity function	*Agglomerative Clustering*									*Partitional Clustering*												
	linkage			*optimizing internal measure*						*repeated bisection*						*direct k-way*					
	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.587	**0.658**	0.645	0.545	0.554	0.513	**0.612**	0.529	0.535	0.544	0.572	0.521	**0.627**	0.541	0.544	0.542	0.573	0.530	**0.613**	0.546	0.547
cor	0.511	0.528	**0.538**	0.521	0.534	0.533	0.418	0.527	**0.540**	0.516	0.526	**0.545**	0.453	0.522	0.536	0.513	0.528	**0.546**	0.472	0.525	0.540
maxen	0.602	0.557	**0.660**	**0.626**	0.615	0.616	0.615	0.570	0.568	0.587	0.591	0.561	**0.609**	0.566	0.566	0.580	0.586	0.561	**0.596**	0.570	0.569
svm	0.603	0.567	**0.647**	**0.644**	0.643	0.614	0.561	0.567	0.561	0.585	**0.596**	0.575	0.586	0.576	0.575	0.575	0.584	0.578	**0.591**	0.570	0.565

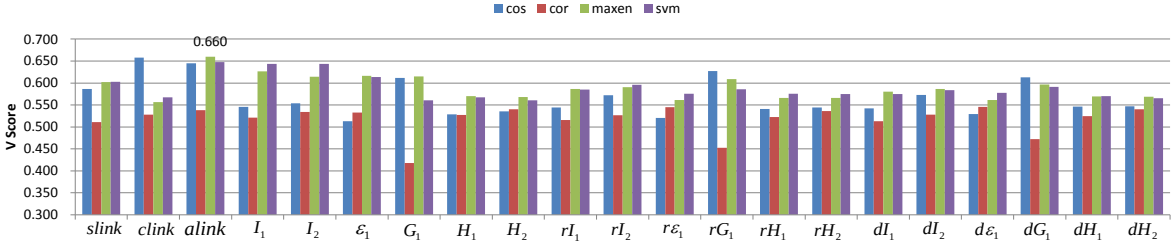


Figure 7.5: Performance of 21 clustering algorithms for all names (known prior K)

similarity function	*Agglomerative Clustering*									*Partitional Clustering*												
	linkage			*optimizing internal measure*						*repeated bisection*						*direct k-way*					
	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.789	**0.829**	0.811	0.718	0.726	0.675	**0.786**	0.692	0.713	0.725	0.745	0.676	**0.790**	0.710	0.717	0.720	0.738	0.697	**0.788**	0.712	0.710
cor	0.708	**0.770**	0.760	0.674	0.676	**0.705**	0.580	0.676	0.691	0.674	0.678	**0.725**	0.603	0.676	0.710	0.672	0.692	**0.720**	0.615	0.676	0.707
maxen	**0.760**	0.660	0.743	0.722	0.722	**0.742**	0.706	0.675	0.678	0.696	0.700	0.675	**0.717**	0.687	0.698	0.688	0.692	0.681	**0.705**	0.701	0.682
svm	**0.861**	0.750	0.834	**0.832**	0.821	0.792	0.729	0.732	0.734	0.769	**0.805**	0.756	0.725	0.764	0.761	0.760	**0.778**	0.748	0.724	0.750	0.746

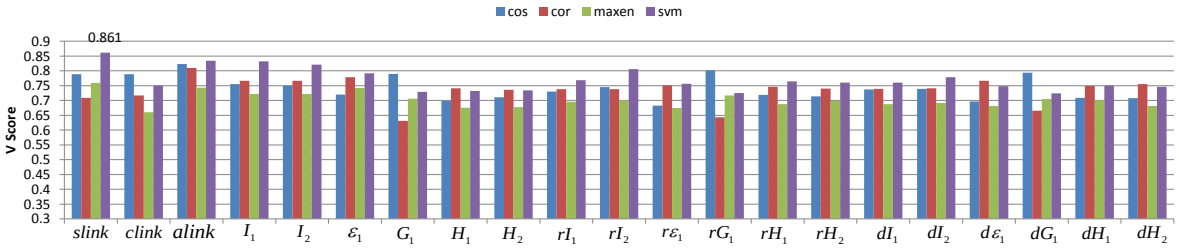


Figure 7.6: Performance of 21 clustering algorithms for person names (known prior K)

similarity function	*Agglomerative Clustering*									*Partitional Clustering*												
	linkage			*optimizing internal measure*						*repeated bisection*						*direct k-way*					
	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.563	0.648	**0.656**	0.501	0.517	0.475	**0.610**	0.488	0.497	0.495	0.539	0.499	**0.632**	0.506	0.515	0.496	0.543	0.507	**0.609**	0.512	0.523
cor	0.472	0.479	**0.505**	0.469	0.487	**0.487**	0.361	0.476	0.486	0.472	0.487	**0.498**	0.416	0.482	0.492	0.466	0.481	**0.501**	0.440	0.483	0.495
maxen	0.606	0.552	**0.694**	**0.649**	0.641	0.627	0.638	0.572	0.568	0.600	0.609	0.571	**0.632**	0.577	0.573	0.594	0.608	0.568	**0.619**	0.575	0.579
svm	0.554	0.550	**0.654**	0.620	**0.639**	0.617	0.568	0.533	0.530	0.552	0.565	0.553	**0.592**	0.547	0.550	0.544	0.555	0.562	**0.601**	0.543	0.542

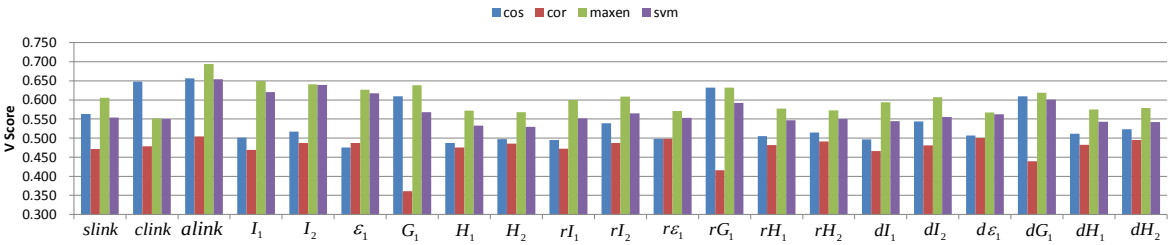


Figure 7.7: Performance of 21 clustering algorithms for organization names (known prior K)

smaller clusters (in many cases, just singleton clusters). Therefore, one-in-one system can perform well if the size the largest cluster is not so large and a number of singleton clusters dominates the distribution, as shown in the example of "Mike Kennedy" in Table 7.9. As the size of the largest cluster increases, V-score of one-in-one system drops as shown in the example of "George Young". The averaged V-score over the 106 names of the one-in-one system is **0.537**. In the later experiments, we will see this is actually a strong baseline.

similarity function	*Agglomerative Clustering*									*Partitional Clustering*												
	linkage			*optimizing internal measure*						*repeated bisection*						*direct k-way*					
	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.417	**0.478**	0.399	**0.483**	0.467	0.441	0.404	0.468	0.447	**0.488**	0.470	0.404	0.408	0.454	0.429	**0.478**	0.470	0.399	0.408	0.460	0.426
cor	**0.403**	0.397	0.377	0.510	0.519	0.476	0.413	0.518	**0.539**	0.467	0.474	**0.484**	0.394	0.471	0.474	0.476	0.485	0.484	0.408	0.481	**0.487**
maxen	0.395	**0.446**	0.440	0.429	0.393	0.425	0.422	0.433	**0.436**	**0.406**	0.394	0.386	0.395	0.380	0.380	**0.401**	0.383	0.391	0.386	0.388	0.395
svm	**0.450**	0.402	0.394	**0.491**	0.438	0.381	0.327	0.481	0.452	**0.472**	0.444	0.429	0.391	0.442	0.431	**0.448**	0.440	0.420	0.396	0.440	0.420

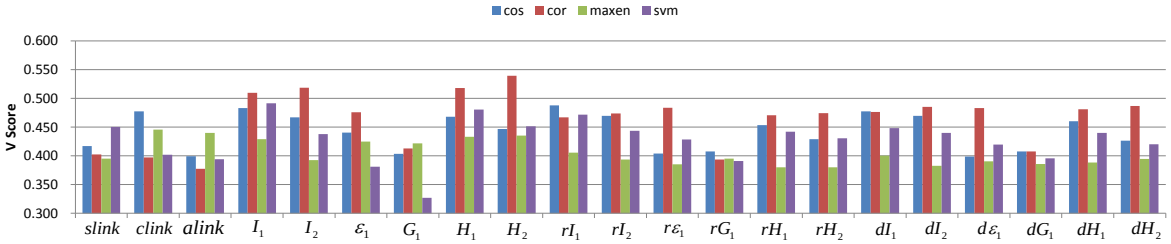


Figure 7.8: Performance of 21 clustering algorithms for GPE names (known prior K)

similarity function	*Agglomerative Clustering*									*Partitional Clustering*												
	linkage			*optimizing internal measure*						*repeated bisection*						*direct k-way*					
	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.520	**0.632**	0.551	0.555	0.557	0.509	**0.561**	0.546	0.538	0.549	0.563	0.507	**0.615**	0.537	0.520	0.549	0.565	0.513	**0.605**	0.534	0.529
cor	0.474	**0.557**	0.515	0.551	0.558	0.556	0.417	0.563	**0.563**	0.556	0.560	**0.565**	0.480	0.554	0.557	0.552	**0.557**	0.555	0.484	0.556	0.555
maxen	0.525	0.493	**0.545**	0.532	0.537	0.537	0.515	0.537	**0.540**	0.536	0.528	0.525	0.498	0.536	0.536	0.531	0.524	0.520	0.510	0.531	0.531
svm	0.511	0.508	**0.552**	0.549	**0.553**	0.528	0.524	0.533	0.534	**0.536**	0.533	0.510	0.525	0.530	0.523	0.530	0.533	0.518	0.532	**0.534**	0.530

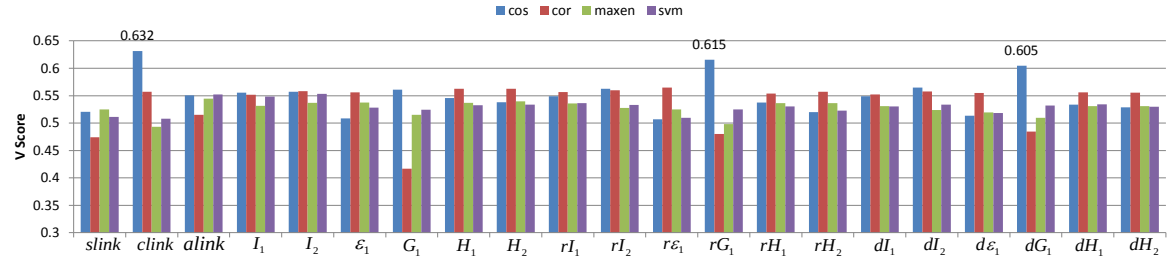


Figure 7.9: Performance of 21 clustering algorithms for all names (unknown prior K)

similarity function	*Agglomerative Clustering*									*Partitional Clustering*												
	linkage			*optimizing internal measure*						*repeated bisection*						*direct k-way*					
	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.681	0.703	**0.733**	**0.718**	0.694	0.678	0.703	0.689	0.656	0.696	0.701	0.626	**0.708**	0.664	0.646	**0.715**	0.711	0.651	0.714	0.684	0.650
cor	0.642	0.601	**0.672**	0.600	0.605	**0.624**	0.486	0.604	0.601	0.587	0.582	**0.592**	0.536	0.589	0.586	0.577	0.587	0.594	0.540	0.593	**0.604**
maxen	0.581	0.557	**0.583**	0.574	0.585	0.590	0.526	0.644	**0.646**	0.603	0.588	0.605	0.534	**0.619**	0.612	0.590	0.584	0.575	0.552	**0.602**	0.587
svm	0.576	0.592	**0.602**	0.573	0.582	0.574	0.521	**0.597**	0.603	0.575	0.583	0.559	0.557	**0.587**	0.569	0.587	0.600	0.595	0.570	**0.618**	0.597

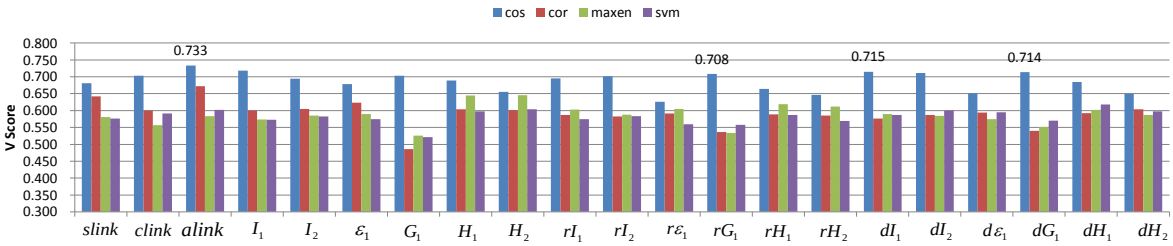


Figure 7.10: Performance of 21 clustering algorithms for person names (unknown prior K)

For all-in-one system, all the instances of each name are clustered into one cluster. The value of *completeness* is 1 because members of a single class are assigned to a single cluster, however, the value of *homogeneity* is 0, because $H(\mathcal{R}|\mathcal{C}) = H(\mathcal{R})$. As a result, the final V-score of the all-in-one system is **0**.

We then tested the performance of the 21 clustering algorithms as described in Section 7.5.2. Figure 7.5, 7.6, 7.7, 7.8 show the performance of 21 clustering algorithms using prior

similarity function	*Agglomerative Clustering*									*Partitional Clustering*												
	linkage			*optimizing internal measure*						*repeated bisection*						*direct k-way*					
	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.480	**0.605**	0.500	0.526	0.531	0.483	**0.527**	0.516	0.512	0.524	0.541	0.485	**0.598**	0.521	0.491	0.521	0.542	0.493	**0.591**	0.514	0.502
cor	0.427	**0.526**	0.486	0.507	0.520	0.525	0.369	**0.526**	0.525	0.521	0.528	**0.534**	0.423	0.521	0.529	0.517	0.526	0.519	0.441	0.526	**0.528**
maxen	0.530	0.505	**0.572**	**0.564**	0.558	0.558	0.549	0.550	0.544	**0.560**	0.559	0.550	0.527	0.563	0.560	0.558	0.554	0.548	0.539	0.560	**0.562**
svm	0.504	0.497	**0.561**	0.553	**0.561**	0.532	0.544	0.523	0.523	**0.541**	0.534	0.503	0.530	0.528	0.524	0.527	0.525	0.505	**0.537**	0.520	0.524

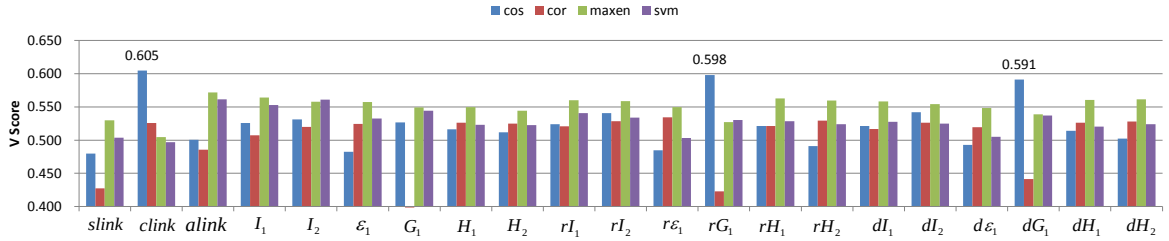


Figure 7.11: Performance of 21 clustering algorithms for organization names (unknown prior K)

similarity function	*Agglomerative Clustering*									*Partitional Clustering*												
	linkage			*optimizing internal measure*						*repeated bisection*						*direct k-way*					
	slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.505	**0.548**	0.515	**0.527**	0.508	0.450	0.522	0.521	0.497	0.524	0.484	0.451	**0.527**	0.496	0.489	**0.519**	0.484	0.442	0.503	0.488	0.481
cor	**0.510**	0.486	0.466	0.538	0.539	0.502	0.447	0.538	**0.547**	**0.523**	0.518	0.515	0.481	0.504	0.499	**0.531**	0.524	0.513	0.475	0.512	0.512
maxen	**0.438**	0.375	0.405	0.368	**0.407**	0.403	0.385	0.360	0.393	**0.370**	0.349	0.343	0.356	0.343	0.363	**0.367**	0.346	0.353	0.358	0.343	0.359
svm	0.457	0.442	**0.461**	**0.504**	0.490	0.457	0.459	0.485	0.486	**0.474**	0.468	0.470	0.465	0.466	0.461	0.469	**0.481**	0.469	0.467	0.477	0.467

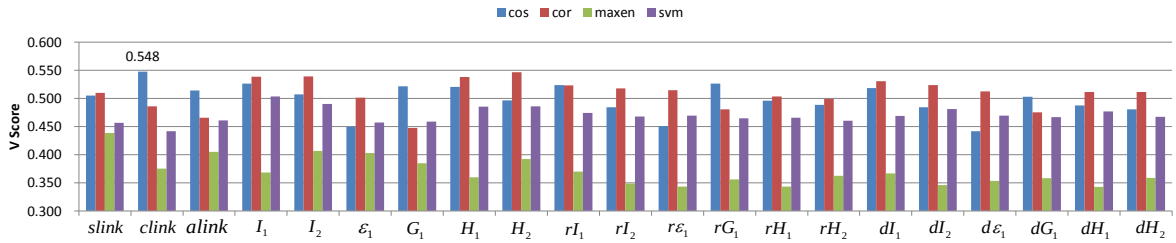


Figure 7.12: Performance of 21 clustering algorithms for GPE names (unknown prior K)

K for all names, person names, organization names, and GPE names respectively. Figure 7.9, 7.10, 7.11, 7.12 show the performance of 21 clustering algorithms without using prior K for all names, person names, organization names, and GPE names respectively. A few top scores are explicitly showed in each figure. Above each figure, there is a table in which the highest score in each category of clustering algorithms (namely, 3 linkage based agglomerative clustering, 6 agglomerative clustering optimizing an internal measure, 6 repeated bisectional partitional clustering and 6 direct k-way clustering) are made bold and underlined, and the highest score among all the scores is grayed.

We first analyze the clustering algorithms using prior K (Figure 7.5, 7.6, 7.7, 7.8).

Observations:

(1) In general, the three linkage based agglomerative clustering algorithms can perform well using any of the four similarity functions. For example, in Figure 7.5 using cosine similarity, the highest score (0.658) is achieved by $clink$, and using maximum entropy based similarity

function, the highest score (0.660) is achieved by $alink$.

(2) In general, the scores for person names are higher than the scores for organization names, and then higher than the scores for GPE names. In other words, in our dataset, the task of clustering person name entities is easier than that of clustering organization name entities, and then easier than that of clustering GPE name entities.

(3) Unlike document clustering, we did not observe the performance advantages of partitional clustering algorithms over agglomerative clustering algorithms, nor did we observe the performance advantages of repeated bisectional clustering algorithms over direct k-way clustering algorithms.

(4) In general, clustering algorithms using correlation (cor) similarity function perform the worst for mixed names (Figure 7.5) and organization names (7.7), however, for GPE names, the highest score is achieved through agglomerative clustering algorithm by optimizing H_2 and applying cor similarity function.

We then analyze the clustering algorithms without using prior K (Figure 7.9, 7.10, 7.11, 7.12). In fact, all of the above four observations still hold except that in general, the scores are relatively lower than those using prior K.

	known K																*unknown K*											
	cos				*cor*				*maxen*				*svm*				*cos*				*cor*				*maxen*				*svm*			
	ALL	PER	ORG	GPE	ALL	PER	ORG	GPE	ALL	PER	ORG	GPE	ALL	PER	ORG	GPE	ALL	PER	ORG	GPE	ALL	PER	ORG	GPE	ALL	PER	ORG	GPE	ALL	PER	ORG	GPE
slink	0	0	0	0	1	1	0	0	0	11	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
clink	18	15	18	0	2	17	1	0	0	0	0	0	0	0	0	0	18	13	18	3	5	4	5	0	0	0	0	0	0	0	0	0
alink	17	2	17	0	3	13	1	0	20	1	19	0	15	9	14	0	0	0	0	0	1	1	2	0	2	0	1	1	8	1	13	0
I_1	0	0	0	0	2	1	1	1	8	1	9	0	16	15	6	4	4	0	1	5	4	1	4	3	0	0	2	0	5	0	8	0
I_2	3	3	4	0	7	2	5	2	3	1	5	0	16	11	15	0	5	0	3	0	4	1	4	3	2	0	0	0	11	0	13	1
$\mathcal{E}_1$	0	0	0	0	5	3	3	1	7	6	1	0	3	4	3	0	0	0	0	0	4	1	4	0	4	1	0	0	0	0	0	0
G_1	6	2	6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	2	0
H_1	0	0	0	0	3	2	1	2	0	0	0	4	0	0	0	2	3	0	0	1	6	1	4	3	1	8	0	0	1	0	0	0
H_2	1	2	1	0	7	3	3	2	0	0	0	3	0	0	0	1	2	0	1	0	5	1	4	4	2	8	0	0	1	0	0	0
rI_1	0	1	0	0	3	1	2	0	0	1	0	0	7	1	2	2	3	0	0	3	4	1	4	1	2	1	0	0	1	0	3	0
rI_2	12	10	11	0	5	2	3	2	9	1	9	0	9	8	4	1	8	6	6	0	4	1	4	0	0	0	0	0	1	0	2	0
$r\mathcal{E}_1$	0	0	1	0	13	11	8	3	0	0	0	0	0	0	0	1	0	0	0	0	6	2	5	0	0	0	0	0	0	0	0	0
rG_1	16	2	16	0	1	0	1	0	1	0	1	0	1	0	1	0	18	11	18	0	1	1	1	0	0	0	0	0	0	0	0	0
rH_1	2	1	1	0	3	2	2	2	0	0	0	0	0	1	0	1	2	0	1	0	4	1	4	0	2	1	0	0	1	0	0	0
rH_2	3	4	3	0	6	4	3	2	0	1	0	0	0	0	0	1	0	0	0	0	4	1	4	0	1	0	0	0	0	0	0	0
dI_1	0	1	0	0	3	2	1	1	0	0	0	0	0	1	0	1	4	0	1	1	4	1	4	2	0	0	0	0	0	0	0	0
dI_2	12	4	11	0	4	3	2	2	0	0	3	0	1	0	1	1	8	9	8	0	4	1	4	1	0	0	0	0	1	0	0	0
$d\mathcal{E}_1$	1	0	1	0	12	3	11	2	0	0	0	0	1	0	2	0	0	0	0	0	4	1	4	0	0	0	0	0	0	0	0	0
dG_1	12	2	11	0	2	0	2	0	0	0	0	0	1	0	1	0	16	7	16	0	1	1	1	0	0	0	0	0	0	0	0	0
dH_1	2	0	2	0	4	3	2	3	0	0	0	0	0	0	0	1	1	0	1	0	4	1	4	0	0	0	0	0	2	3	0	0
dH_2	3	1	6	0	9	3	5	3	0	0	0	0	0	0	0	0	1	0	0	0	4	1	4	0	1	0	0	0	1	0	0	0

Table 7.10: Comparison of 21 clustering algorithms

It is worth noting that each score in the above figures is the average of scores over all the names and from the figures, we can not see whether one score is statistical significant than the others. In fact, like document clustering, we can also create significance matrices among 21 clustering algorithms, however, the number of such significance matrices could reach 32 (8 figures * 4 similarity functions in each figure). Tabulating such matrices in this thesis not only

occupies lots of space but also makes it difficult to interpret what are good clustering algorithms in general. To overcome this problem, we present a matrix which summarizes the results among the 32 significance matrices as shown in Table 7.10. The rows in the table are the 21 clustering algorithms. The columns are mainly split into two categories (known K and unknown K), and then the columns are further split according to the four similarity functions, and finally, each column of similarity function is further split by four cases, namely, all names (ALL), person names (PER), organization names (ORG) and GPE names (GPE). In total, there are 32 fine-grained columns. Each number in the cell represents the number of clustering algorithms that are statistically significantly worse than the algorithm in the row. The higher the number is, the better the clustering algorithm is. The maximum possible number is 20 which means the algorithm can statistically significantly outperform all the other 20 algorithms. In this way, we can clearly observe which algorithms are normally better than the others in general. We highlight those numbers in the table which indicate that the algorithm can outperform at least half of the other clustering algorithms ($>= 12$). From the table, we can easily observe some important facts.

Observations:

(1) $clink$ is a good clustering algorithm for mixed names (outperform the other 18 algorithms), person names (15), and organization names (18) by applying cosine similarity function. This observation holds for either known K or unknown K.

(2) $alink$ is a good algorithm for mixed names (17) and organization names (17) by applying cosine similarity function. This observation only holds for known K and does not hold for unknown K.

(3) rG_1 is a good algorithm for mixed names (17) and organization names (17) by applying cosine similarity function. This observation holds for either known K or unknown K.

(4) if correlation similarity function is applied, $clink$ and $alink$ are good ones for only person names. This observation only holds for known K and does not hold for unknown K.

(5) if maxen or svm similarity function is applied, $alink$ is a good one for mixed names and organization names. This observation only holds for known K and does not hold for unknown K.

(6) there is not a clustering algorithm that can statistically significantly perform better than the others for GPE names. Recall that we obtained relatively lower scores for GPE names which clearly shows that clustering GPE name entities is harder than clustering person or organization name entities.

The further analyses of the above observations are discussed as follows.

Analyses:

(1) the "long tail effect" discussed in Section 7.5.4 tells that our dataset has a very unbalanced class distribution. Like document clustering, we can compute the the coefficient of variation (CV) to measure how unbalanced a clustering is. Similarly, we can compute the CV value of true clustering which is denoted as CV_0, and then we can compute the CV value of a system (CV_1) clustering generated by any clustering algorithm. For a good clustering algorithm, the cluster distribution should be as close as the true class distribution, i.e., $DCV = CV_1 - CV_0$ should be close to 0 as much as possible. A clustering algorithm which produces a balanced cluster distribution normally lead to a small CV_1, therefore, DCV is normally negative.

The results are shown in Table 7.11 and Table 7.12, in which except the column of "CV_0", the number in each cell represents the DCV value of each specific clustering algorithm. We can clearly see that the DCV values of $clink$, $alink$, rG_1 are closer to 0 than most of the others when cosine similarity function is applied. This tells why $clink$, $alink$ and rG_1 can produce better results from the aspect of class distribution.

similarity function	CV_0	*Agglomerative Clustering*									*Partitional Clustering*												
		linkage			*optimizing internal measure*						*repeated bisection*						*direct k-way*						
		slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.85	0.28	-0.10	0.12	-0.33	-0.43	-0.57	-0.07	-0.46	-0.50	-0.35	-0.48	-0.61	-0.12	-0.49	-0.56	-0.36	-0.48	-0.61	-0.14	-0.50	-0.57
cor	0.85	0.33	-0.10	0.18	-0.26	-0.27	-0.32	-0.59	-0.27	-0.28	-0.30	-0.31	-0.35	-0.70	-0.31	-0.33	-0.31	-0.32	-0.36	-0.70	-0.33	-0.35
maxen	0.85	0.22	-0.18	0.01	-0.20	-0.20	-0.24	-0.16	-0.38	-0.40	-0.35	-0.34	-0.42	-0.26	-0.38	-0.39	-0.36	-0.35	-0.43	-0.30	-0.38	-0.40
svm	0.85	0.23	-0.18	0.04	-0.19	-0.18	-0.25	-0.17	-0.40	-0.38	-0.34	-0.32	-0.40	-0.27	-0.37	-0.38	-0.35	-0.37	-0.41	-0.31	-0.40	-0.41

Table 7.11: DCV values of 21 clustering algorithms for all names (known K)

similarity function	CV_0	*Agglomerative Clustering*									*Partitional Clustering*												
		linkage			*optimizing internal measure*						*repeated bisection*						*direct k-way*						
		slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0.85	0.11	-0.12	0.03	-0.28	-0.36	-0.51	-0.04	-0.41	-0.44	-0.30	-0.44	-0.59	-0.12	-0.49	-0.54	-0.32	-0.44	-0.58	-0.17	-0.49	-0.53
cor	0.85	0.14	-0.08	0.05	-0.26	-0.26	-0.29	-0.56	-0.28	-0.28	-0.29	-0.27	-0.31	-0.68	-0.30	-0.29	-0.31	-0.29	-0.34	-0.69	-0.33	-0.31
maxen	0.85	0.10	-0.26	-0.08	-0.25	-0.23	-0.26	-0.21	-0.35	-0.36	-0.35	-0.37	-0.39	-0.34	-0.38	-0.38	-0.36	-0.38	-0.41	-0.36	-0.37	-0.39
svm	0.85	0.12	-0.22	-0.04	-0.23	-0.22	-0.24	-0.19	-0.35	-0.36	-0.33	-0.35	-0.41	-0.32	-0.37	-0.38	-0.33	-0.37	-0.42	-0.31	-0.37	-0.38

Table 7.12: DCV values of 21 clustering algorithms for all names (unknown K)

(2) As we have observed, $clink$ is a good clustering algorithm for name entity clustering. We illustrate by some examples and explain why it can perform extremely well in some cases and why it fails in some other cases.

First, let us start from the "long tail effect" in our dataset, which shows that for many names, the gold clustering has a few large clusters and many small clusters (or even singletons). We have observed that in many cases, most of instances in those large clusters are tightly connected (i.e., similarity values of pairwise instances are high), therefore, usually any clustering algorithm has no difficulty in putting them in one cluster, however, some instances from those large clusters look more like from another cluster since the similarity values between them and other tightly connected instances are relatively low. The key issue for any clustering algorithm is whether it can correctly place those "fake" outliers into the cluster which already has some tightly connected instances, and meanwhile it can correctly place the true outliers in small clusters or singletons.

Second, let us analyze the characteristics of some clustering algorithms. For $slink$, the similarity of two clusters is defined as the largest pairwise similarity between members of the two clusters. For $clink$, the similarity of two clusters is defined as the smallest pairwise similarity, while for $alink$, the similarity of two clusters is defined as the average pairwise similarity. As a result, $slink$ is prone to confuse the "fake" outliers and the true outliers, while $clink$ and $alink$ are more likely to correctly distinguish them. We use the following example to illustrate this problem. For those clustering algorithms that optimize an internal measure, some of the properties have been discussed in Chapter 6.

We then use the following example to help analyze why some algorithms have advantages.

Figure 7.13 shows an example of clustering 5 instances with the name of "National Students Union". From the clustering plot, we can clearly observe how instances are clustered during each step of hierarchical clustering. We can observe that the three linkage based agglomerative clustering together with G_1 can produce a perfect clustering, while I_1,I_2,H_1,H_2 behave similarly by incorrectly put instance 1 into the cluster of 2,3, ε_1 also makes a mistake by clustering instance 1 with 3.

(3) As we have observed, the performance of clustering algorithms also depends on the applied similarity function. According to the 8 figures (Figure 7.5-Figure 7.12), in general, we did not observe that the two supervised similarity functions (maxen, svm) can significantly perform better than the two unsupervised similarity functions (cos, cor). As we can observe in

Name: National Students Union Instances: {1,2,3,4,5} Gold clustering: {{1},{2,3,4,5}}
Similarity matrix computed by cosine similarity:

1.0000	0.0108	0.0144	0.0247	0.0237
0.0108	1.0000	0.0671	0.1610	0.1518
0.0144	0.0671	1.0000	0.0480	0.0731
0.0247	0.1610	0.0480	1.0000	0.9463
0.0237	0.1518	0.0731	0.9463	1.0000

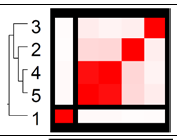
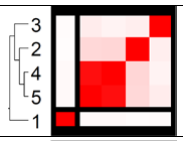
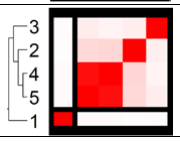
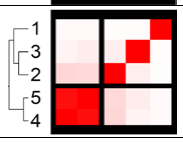
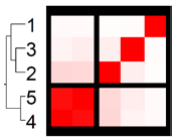
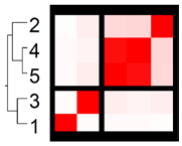
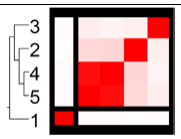
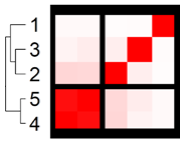
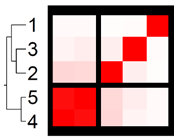
clustering algorithm	system clustering	clustering plot	clustering algorithm	system clustering	clustering plot
slink	{{1},{2,3,4,5}} V-score=1.0		*clink*	{{1},{2,3,4,5}} V-score=1.0	
alink	{{1},{2,3,4,5}} V-score=1.0		I_1	{{1,2,3},{4,5}} V-score= 0.202	
I_2	{{1,2,3},{4,5}} V-score= 0.202		$\mathcal{E}_1$	{{1,3},{2,4,5}} V-score= 0.38	
G_1	{{1},{2,3,4,5}} V-score=1.0		H_1	{{1,2,3},{4,5}} V-score= 0.202	
H_2	{{1,2,3},{4,5}} V-score= 0.202				

Figure 7.13: An example for comparison of clustering algorithms

Figure 7.7 and Figure 7.11, to cluster organization name entities (either known K or unknown K), cor related results are significantly worse than the other three. In Figure 7.12, to cluster GPE name entities (unknown K), maxen related results are significantly worse than the other three. Again the 8 figures do not show whether some similarity function can statistically significantly perform better than the others. Therefore, we make a significance table for similarity functions as shown in Table 7.13.

In the table, the rows represent the four similarity functions, the columns are split into two major parts, known K and unknown K, and then further split into 4 parts, All, PER, ORG and GPE, and finally, the rows and columns form 8 4*4 significance matrices. Each cell is a number tells how many clustering algorithms (21 at most) one similarity function can statistically significantly perform better than the other. The larger the number is, the better the similarity function is. The possible largest number is 21, which indicates the similarity

function can consistently work better than the other using any of the 21 clustering algorithms. We highlight the numbers which is larger than at least half of 21 (> 11) in the table.

The table clearly shows that cos similarity function has no significant difference from maxen or SVM. For either ALL or ORG (known K), cor similarity function is significantly worse than the other three. For GPE (unknown K), maxen is significantly worse.

	known K																*unknown K*															
	ALL				*PER*				*ORG*				*GPE*				*ALL*				*PER*				*ORG*				*GPE*			
	cos	cor	maxen	svm	cos	cor	maxen	svm	cos	cor	maxen	svm	cos	cor	maxen	svm	cos	cor	maxen	svm	cos	cor	maxen	svm	cos	cor	maxen	svm	cos	cor	maxen	svm
cos	–	12	1	3	–	6	1	0	–	13	1	1	–	0	0	0	–	4	3	3	–	4	5	5	–	4	2	2	–	0	9	0
cor	0	–	0	0	0	–	1	0	0	–	0	0	0	–	0	0	4	–	1	2	0	–	6	3	1	–	0	0	0	–	14	0
maxen	4	17	–	0	0	3	–	0	16	21	–	1	0	0	–	0	0	1	–	0	0	0	–	0	5	10	–	0	0	0	–	0
svm	11	20	0	–	3	14	12	–	11	20	0	–	0	0	0	–	0	1	0	–	0	0	0	–	0	4	0	–	0	0	15	–

Table 7.13: Comparison of similarity functions

(4) Determining the number of clusters K is a hard issue for any clustering problem without using the prior K, and as we have observed, clustering algorithms without using prior K normally perform worse than clustering algorithms using prior K. In order to study the impact of Algorithm 5 in Chapter 5, we compared the system generated number of clusters k_1 with the true number of classes k_0, and computed the percentages of whether the algorithm can produce

- a good k_1 ($k_1 - k_0 = 0$ or ± 1)
- a mediocre k_1 ($k_1 - k_0 = \pm 2$ or ± 3)
- a bad k_1 (others)

The results are shown in Table 7.14 in which each number shows the percentage that Algorithm 5 in Chapter 5 can produce a good or a mediocre or a bad number of clusters, for example, when the clustering algorithm $clink$ with cosine similarity function is applied, for 41.7% of the 106 names in our dataset, the algorithm can produce a good number of clusters (equal to or just one more or one fewer than the true number of classes). Table 7.14 shows that when the cosine similarity function is applied, $clink$, $alink$, rG_1 and dG_1 have higher percentages of producing good or mediocre k_1. We have known that those few clustering algorithms can perform better than the others using prior K, and now we also know that better k_1 can be obtained for those algorithms. Taking these two factors into considerations, it is easier to understand why we also obtained higher performance for those algorithms without using prior K.

similarity function	*range* k_1-k_0	*Agglomerative Clustering*									*Partitional Clustering*											
		linkage（%）			*optimizing internal measure*（%）						*repeated bisection*（%）						*direct k-way*（%）					
		slink	*clink*	*alink*	I_1	I_2	$\mathcal{E}_1$	G_1	H_1	H_2	rI_1	rI_2	$r\mathcal{E}_1$	rG_1	rH_1	rH_2	dI_1	dI_2	$d\mathcal{E}_1$	dG_1	dH_1	dH_2
cos	0,±1	40.6	41.7	41.7	25	27.1	22.9	41.7	27.1	27.1	28.1	30.2	27.1	39.6	33.3	28.1	29.2	32.3	27.1	41.7	35.4	32.3
	±2,±3	20.8	21.9	24	24	26	22.9	25	28.1	25	21.9	22.9	26	25	24	24	24	22.9	24	21.9	22.9	20.8
	others	38.5	36.5	34.4	51	46.9	54.2	33.3	44.8	47.9	50	46.9	46.9	35.4	42.7	47.9	46.9	44.8	49	36.5	41.7	46.9
cor	0,±1	43.8	45.8	44.8	42.7	42.7	41.7	37.5	42.7	41.7	44.8	43.8	47.9	36.5	43.8	43.8	45.8	47.9	42.7	34.4	44.8	50
	±2,±3	20.8	22.9	19.8	20.8	24	26	21.9	20.8	24	18.8	20.8	21.9	28.1	19.8	22.9	24	20.8	25	33.3	27.1	20.8
	others	35.4	31.2	35.4	36.5	33.3	32.3	40.6	36.5	34.4	36.5	35.4	30.2	35.4	36.5	33.3	30.2	31.2	32.3	32.3	28.1	29.2
maxen	0,±1	55.2	47.9	59.4	57.3	58.3	54.2	56.2	53.1	55.2	57.3	57.3	57.3	59.4	58.3	59.4	57.3	57.3	60.4	55.2	58.3	61.5
	±2,±3	20.8	27.1	26	21.9	20.8	25	20.8	22.9	20.8	22.9	21.9	17.7	25	20.8	19.8	24	21.9	17.7	22.9	19.8	15.6
	others	24	25	14.6	20.8	20.8	20.8	22.9	24	24	19.8	20.8	25	15.6	20.8	20.8	18.8	20.8	21.9	21.9	21.9	22.9
svm	0,±1	37.5	40.6	43.8	51	45.8	51	41.7	42.7	42.7	47.9	41.7	44.8	50	45.8	46.9	50	45.8	44.8	49	45.8	44.8
	±2,±3	32.3	28.1	30.2	25	29.2	24	30.2	29.2	26	25	27.1	20.8	27.1	25	21.9	22.9	24	26	20.8	27.1	25
	others	30.2	31.2	26	24	25	25	28.1	28.1	31.2	27.1	31.2	34.4	22.9	29.2	31.2	27.1	30.2	29.2	30.2	27.1	30.2

Table 7.14: Comparison of system generated K and prior K

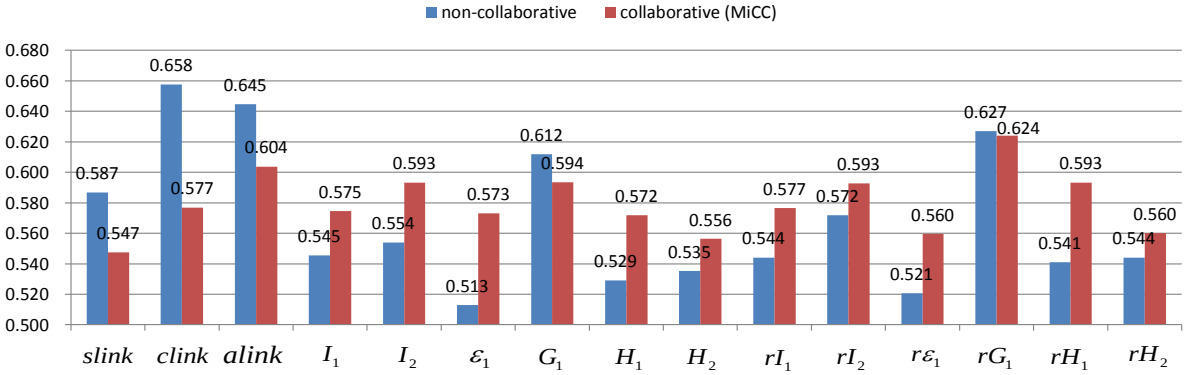


Figure 7.14: Impact of MiCC (known K)

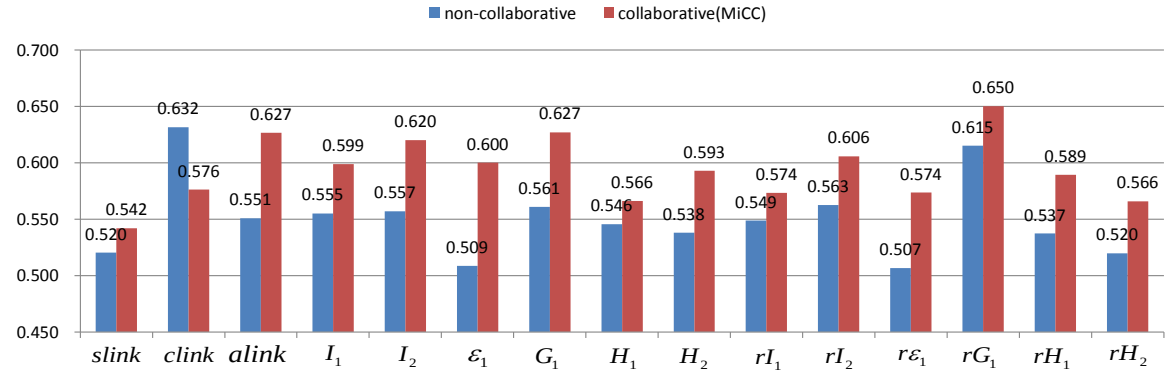


Figure 7.15: Impact of MiCC (unknown K)

7.5.4.2 Impact of MiCC

For each name, we first applied Lucene[1] to search at most 100 documents that mention the name and use them as the pool of collaborative instances. It is worth noting that the selected instances are actually ranked by the default ranking model in the Lucene. Similar with document clustering, we selected Silhouette Coefficient (*SC*) as our internal measure. For each iteration, we selected at most $n_{step} = 5$ collaborative instances. To pick the best 5 collaborative documents in each iteration, we tried $n_{trials} = 10$ times by randomly selecting 5 documents

[1]http://lucene.apache.org/

from the pool. For each trial, we added the 5 documents into the expanded set of documents, and applied a clustering algorithm on the newly expanded set of document and then computed the clustering quality of the original set of instances using the internal measure *SC*. Among the 10 trials, we picked the best 5 collaborative documents which led to the best clustering quality. We set $n_{iterations} = 6$, in other words, we at most selected 30 collaborative documents in the end.

We tested on the 15 baseline clustering algorithms ($alink$,$clink$,$alink$,I_1, I_2, ε_1, G_1, H_1, H_2, rI_1, rI_2, $r\varepsilon_1$, rG_1, rH_1, rH_2) , and validated whether the MiCC algorithm can help each baseline algorithm to achieve better performance evaluated by V-measure. Figure 7.14, 7.15 show the results for known K and unknown K respectively.

We observed that MiCC achieves limited success for some clustering algorithms while fails for some other clustering algorithm if the number of clusters (K) is given. There are two reasons that lead to the possible failure: (1) in name entity clustering, the cluster distribution in instance collaborators are still quite close to the original set of instances. Therefore the added collaborators can only enhance some already well-formed clusters in the original dataset and still lack the ability to help uncover bad clusters; (2) the added collaborators may belong to a new cluster that is not in the original dataset. Thus if K is set to be fixed, those new added collaborators which belong to a new cluster will be distributed to clusters in the original dataset, thus they are introduced as noises which make the results worse.

However, if K is not given, MiCC works for most of the clustering algorithms. That clearly shows the approach that automatically determines K helps to reduce the side-effect of introducing new clusters as K can be dynamically changed.

7.5.4.3 Impact of MaCC

We collected 84 clustering results, which is a combination of 21 clustering algorithms and 4 similarity functions.

Similar with what we have done in document clustering, we experimented four combination schemes and four consensus functions, specifically,

Scheme 1. The 84 clustering results are categorized into 4 serial groups, the first group

uses cosine similarity, the second group uses correlation similarity, the third group uses maxen similarity and the final group uses svm similarity. Each group contains 21 clustering results. We use notations "cos", "cor", "maxen" and "svm" to represent the four groups.

Scheme 2. The 84 clustering results are categorized into 3 serial groups, the first group includes 24 repeated bisectional clustering results (6 repeated bisectional clustering algorithms times 4 similarity functions), the second group includes 24 direct k-way clustering results (6 direct k-way clustering algorithms times 4 similarity functions) and the third group includes 36 agglomerative clustering results(9 agglomerative clustering algorithms times 4 similarity functions), . We use notations "rbr", "direct" and "agglo" to represent the three groups respectively.

Scheme 3. The 84 clustering results are ranked from the highest to the lowest by the scores computed by the internal measure "silhouette coefficient", and then split into 4 groups, each group containing 21 ranked clustering results.

Scheme 4. The 84 clustering results are ranked from the highest to the lowest by the scores computed by the external measure "V-measure", and then split into 4 groups, each group containing 21 ranked clustering results.

The four consensus functions include co-association matrix based, IBGF, CBGF, and HBGF.

Figure 7.16 and Figure 7.17 show the performance of the four combination schemes and four consensus functions for known K and unknown K respectively. We can observe that:

(1) In Figure 7.16 (a), after cor related clustering results are added, the performance significantly drops which clearly show that they hurt the performance. However, after adding maxen and svm related clustering results, the performance increases again. The final best performance is obtained after adding all the 84 clustering results, however, the score 0.646 cannot beat the best one (0.660) among the 84 clustering results although it beats 95.2% of 84. This observation does not quite hold for unknown K in which case, cor related clustering results are significantly worse than the others, and the best combination performance (0.621) is obtained only by adding cosine related clustering results and it also cannot beat the best score among the 84 clustering results (unknown K) which is 0.632.

(2) In Figure 7.16 (b), we do not observe significant improvement after adding direct related clustering results which is consistent with the fact that direct related results are not sig-

nificantly better than rbr, however, after adding agglomerative related results, the performance can increase significantly, because we already know that three linkage based agglomerative clustering algorithms perform well. This observation also holds for unknown K.

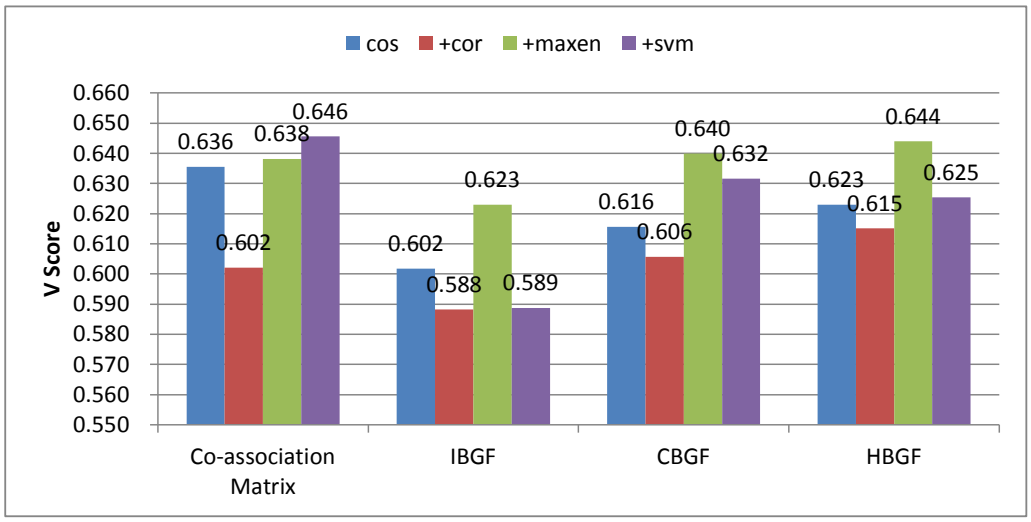
(3) Better performance can be obtained by Scheme 3 and Scheme 4 either for known K or unknown K. Most importantly, the best combined score can all outperform the best one in the 84 clustering results. For known K, in Figure 7.16 (c), we obtained the best score of 0.699, in Figure 7.16 (d), we obtained the best score of 0.750. They outperform the best individual score of 0.660 by 3.9% and 9.0% respectively. For unknown K, in Figure 7.17 (c), we obtained the best score of 0.646, in Figure 7.17 (d), we obtained the best score of 0.751. They outperform the best individual score of 0.632 by 1.4% and 11.9% respectively. All those performance gains are statistically significant.

Similar to the conclusions we obtained from document clustering, we claim that

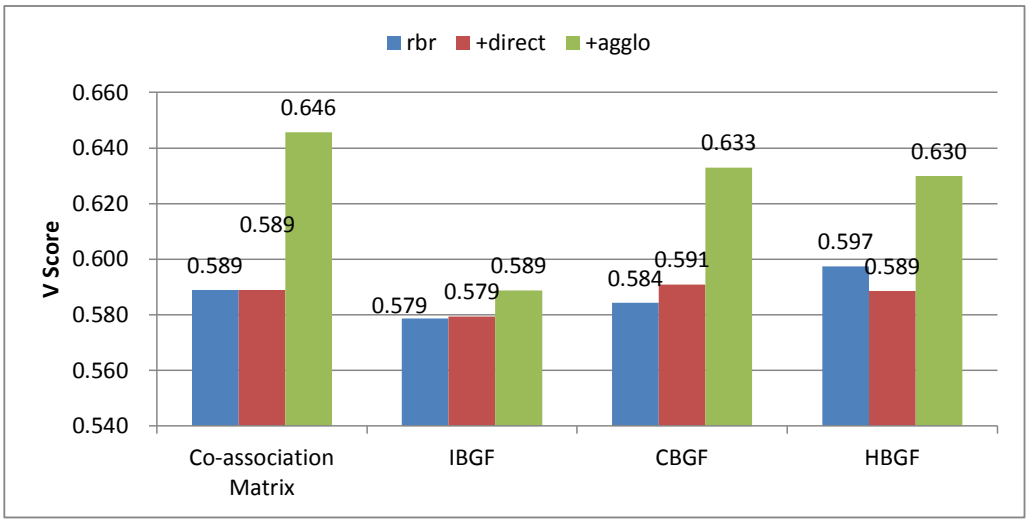
(1) the success of MaCC relies on clustering diversity. For example, direct related clustering results perform comparably with rbr related clustering results in Figure 7.16 (b), therefore, direct related results do not have enough diversity to further improve the combined score.

(2) the success of MaCC relies on selection of clusterers. Good clusterers tend to produce even better final result, while bad clusterers tend to produce negative result. This is obvious from Figure 7.16 (c) and (d).

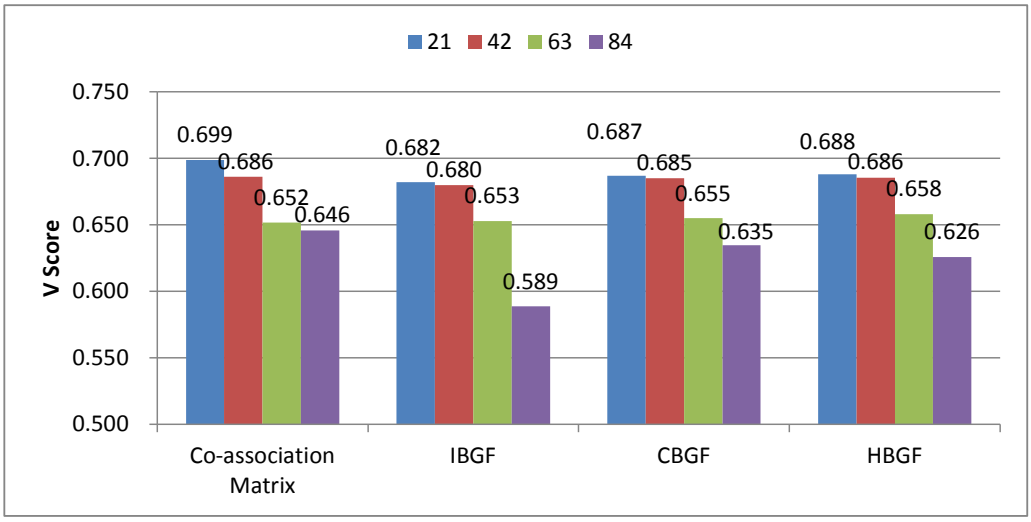
(3) the success of MaCC also relies on consensus function. Co-association matrix based is a good choice according to our experiments.



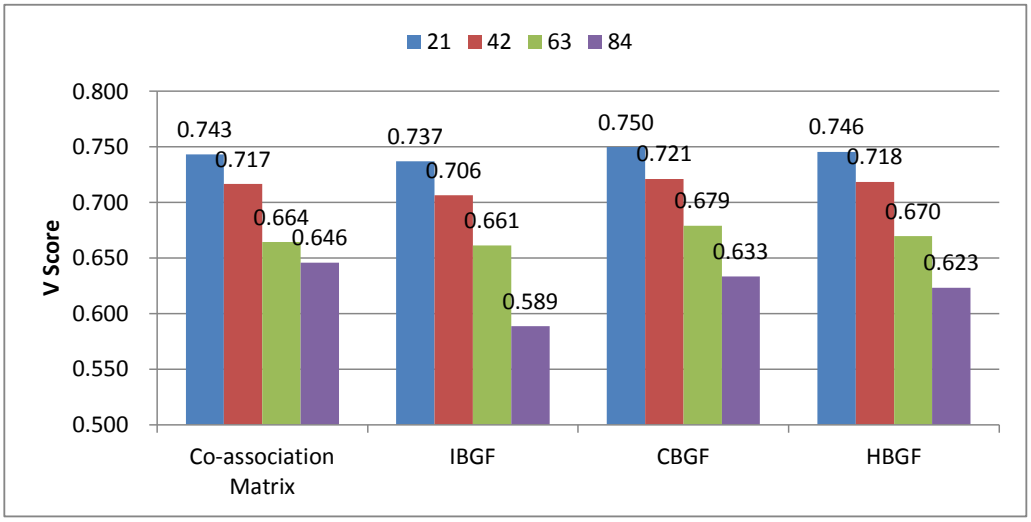
(a) Scheme 1: combination by series of similarity functions (known K)



(b) Scheme 2: combination by series of clustering algorithms (known K)

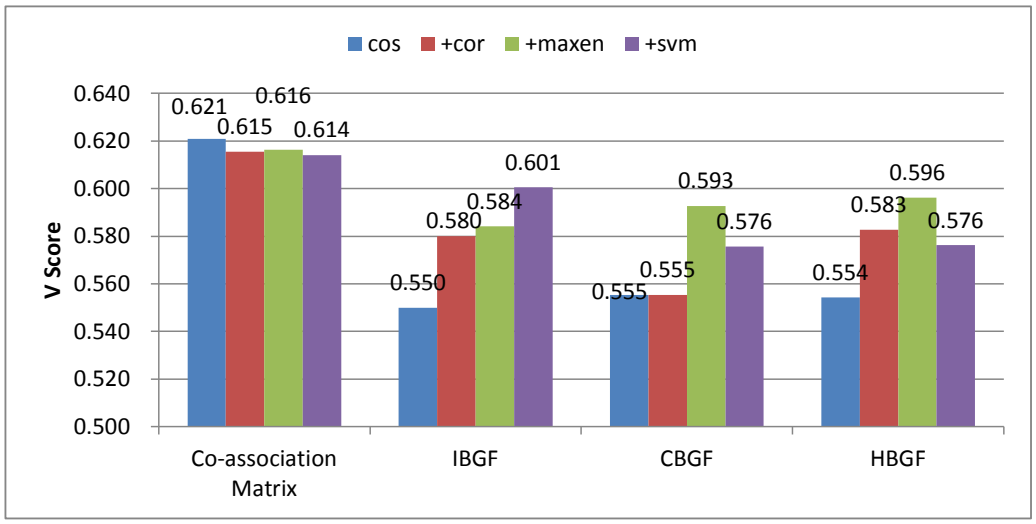


(c) Scheme 3: combination by ranks of silhouette coefficient (known K)

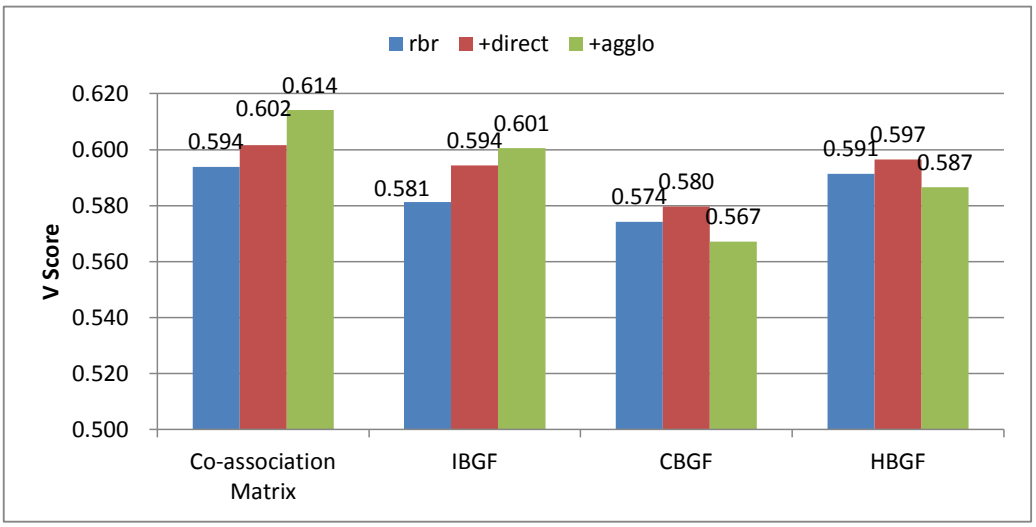


(d) Scheme 4: combination by ranks of V measure (known K)

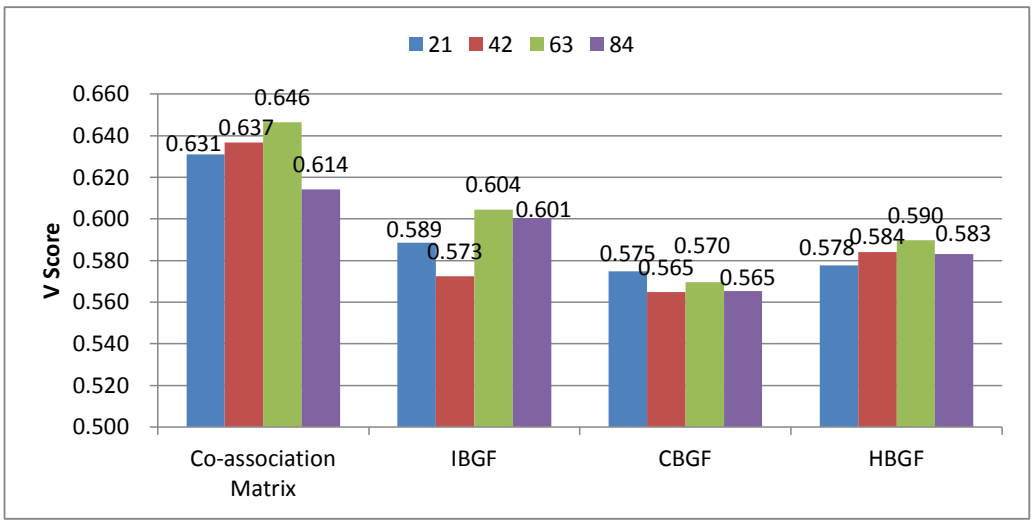
Figure 7.16: Four combination schemes of MaCC (known K)



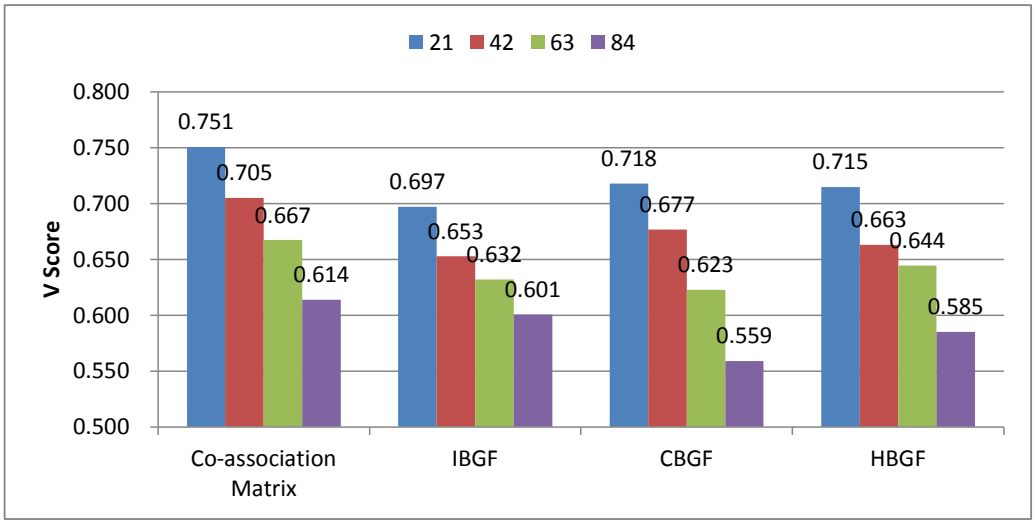
(a) Scheme 1: combination by series of similarity functions (unknown K)



(b) Scheme 2: combination by series of clustering algorithms (unknown K)



(c) Scheme 3: combination by ranks of silhouette coefficient (unknown K)



(d) Scheme 4: combination by ranks of V measure (unknown K)

Figure 7.17: Four combination schemes of MaCC (unknown K)

7.6 Summary

In this chapter,

(1) we have implemented a rule based classifier to identify name variation. We extensively studied the two types of errors, and claim that in order to improve the performance, we can either collect more resources or design more complicated rules.

(2) we have studied feature contributions in name disambiguation classification models. For different entity types, different features weigh differently, and separate models for entity types can perform better than a single model without distinguishing entity types.

(3) we have identified two types of "long tail effect" in name entity clustering. We extensively studied the performance of various baseline clustering algorithms, discussed why we only achieved limited success in MiCC, and finally, we validated the effectiveness of MaCC.

Chapter 8

Conclusions and Future Directions

8.1 Conclusions

In this thesis,

- we proposed two collaborative learning schemes, i.e., collaborative ranking (CR) and collaborative clustering (CC). Each of the two collaborative learning schemes leverages two levels of collaboration and the combination of the two levels.

  More specifically, collaborative ranking seeks (1) query-level collaboration (MiCR) which searches query collaborators and leverages the redundant and complementary information in those query collaborators to help improve ranking decisions; (2) ranker-level collaboration (MaCR) which combines the ranking decisions from multiple rankers. The query-level collaborative ranking (micro collaborative ranking) should not be confused with the traditional query expansion technique which is mostly referred to as query reformulation and there are no such functionalities of leveraging multiple helpful collaborators of the query. The ranker-level collaborative ranking (macro collaborative ranking) corresponds to some other proposed names such as "consensus ranking", "ensemble ranking" in the community. The reason why we assign a new name here is to make it a counterpart of "query-level" and "micro". To summarize, there are two innovations in Collaborative Ranking: (1) the proposed concept of collaborative queries and the mechanism of combining multiple collaborative queries. (2) the first trial of combining query level

collaborative ranking and ranker-level collaborative ranking (MiMaCR).

Collaborative clustering seeks (1) instance-level collaboration (MiCC) which searches instance collaborators which can potentially help recover the clustering structure hidden in the original data; (2) clusterer-level collaboration (MaCC) which combines the clustering results from multiple clusterers. The instance-level collaborative clustering is a novelty proposed in this thesis while the clusterer-level collaboration corresponds to some other known names such as "consensus clustering" or "ensemble clustering" (again, the reason why we make a new name here is to make it a perfect counterpart of "instance-level" and "micro"). Another novelty of this thesis is the first trial of combining instance-level collaborative clustering and clusterer-level collaborative clustering (MiMaCC).

Actually the same similar scheme can be proposed for classification problems, i.e., for a testing sample in a classification problem, the collaborative classification seeks (1) sample-level collaboration which searches other potential helpful samples that can provide redundant or complementary information which can then be combined to help classification; (2) classifier-level collaboration which combines the classification results from multiple classifiers. Similarly, sample-level can be combined with classifier-level to achieve even better results.

A more general scheme which can cover the above three types of machine learning problems (classification, ranking and clustering) is obvious which seeks (1) object-level collaboration in which object means testing sample in classification, query in ranking or instance in clustering; (2) model-level collaboration in which model can mean classifier for classification, ranker in ranking or clusterer in clustering. The combination of object-level and model-level still works.

- we studied how MiCR, MaCR and MiMaCR can be applied in entity linking problem, and showed

  (1)there are several important factors that determine whether MiCR can succeed including the searching strategy of query collaborators (not all potential query collaborators are helpful, and noisy query collaborators can actually hurt the result) and the combination

scheme of multiple query collaborators.

(2)there are also several important factors that determine whether MaCR can succeed including the diversity of multiple rankers, the combination scheme of multiple ranking results.

(3)MiMaCR can achieve better results than either MiCR and MaCR.

- we studied various factors involved in document clustering, including comparison of various external measures, four similarity functions, 21 clustering algorithms and 12 internal measures. Mostly importantly, we validated the effectiveness of MiCC, MaCC and MiMaCC on this application. We showed that

  (1) collaborative instances are really helpful to reconstruct the clustering structure if the structure in the original dataset is implicit and not easy for any baseline clustering algorithm to identify.

  (2) there are also several important factors that determine whether MaCC can succeed : diversity of multiple clusterers, the order of adding multiple clusterers (selection scheme), and consensus function (the approach to obtain a consensus clustering result based on multiple clustering results).

- we studied two key challenges in name entity clustering, including name variation and name disambiguation.

  (1) For name variation, more person, organization and GPE related resources should be collected for further performance improvement.

  (2) For name disambiguation, global (document-wide) features and GPE related features have more contributions to disambiguate ORG or GPE names, while local (context surrounding the target entity) features have more contributions to disambiguate PER names.

  (3) MiCC achieves limited success for some clustering algorithms while fails for some other clustering algorithm if the number of clusters (K) is given. It clearly shows that searching collaborative instances for name entity clustering is much harder than document clustering as the added collaborators may still lack the ability to help recover clus-

tering structure or even worse introduce noises that should be excluded from the original K clusters. If K is not given, MiCC works for most of clustering algorithms. That clearly shows the approach that automatically determines K helps to reduce the side-effect of introducing new clusters (as K can be dynamically changed).

(4) The several factors that determine whether MaCC can succeed still apply in name entity clustering.

8.2 Future Directions

8.2.1 Coupling Collaborative Ranking and Collaborative Clustering

Ranking and clustering are normally considered as two different problems and they do not have tight connections. Recently, Sun et al. [86] addressed a problem of generating clusters for a specified type of objects, as well as ranking information for all types of objects based on these clusters in a heterogeneous information network. The motivation is twofold: first, it often leads to unwanted results by ranking objects globally without considering which clusters they belong to, for example, globally ranking papers from two quite different areas does not make much sense; second, it also leads to dull results by clustering a huge number of objects without differentiating the ranks of objects, for example, it makes no sense to cluster thousands of reviews of a restaurant without first filtering worthless (lower-ranked) reviews (e.g., advertisement). A novel framework RankClus they proposed can mutually enhance the quality of clustering and ranking until little change can be made.

In fact, in this thesis, we already coupled the usage of ranking and clustering either in collaborative ranking or in collaborative clustering. For collaborative ranking, the main method of searching good collaborators is achieved by clustering. For collaborative clustering, when searching collaborative instances in the application of name entity clustering, we use Lucene search engine to produce a pool of ranked collaborative instances and when considering the combination order of multiple clustering results, we also tried ranking method either by internal measure or external measure.

We can further enhance the connection of ranking and clustering from the following aspects:

(1) when searching collaborative queries in collaborative ranking, we can apply MaCC to produce a more robust clustering.

(2) when searching collaborative instances in collaborative clustering, we can apply MaCR to produce a more robust ranking.

8.2.2 Enhancing MiCR

In Chapter 3, although we proposed a weighted version of combining the contributions of diverse query collaborators, we did not elaborate how we can compute such weights. There are two promising approaches and the effectiveness should be validated using real applications such as entity linking.

Approach 1: Rank by distance to the cluster center.

The closer the collaborators are to the center of the cluster, the more important the collaborators are. Let C represent the cluster that contain the query q and its k query collaborators $cq_1, ..., cq_k$, i.e., $C = \{q, cq_1, ..., cq_k\}$. Each mention in the cluster is a d-dimensional vector. We use the same notation q, cq_1,...,cq_k to represent the vectors. Then the centroid of the cluster can be computed as:

$$centroid = \frac{1}{k+1}(q + \sum_{i=1}^{k} cq_i)$$

The distance from each query collaborator to the centroid can be computed using Euclidean distance, Mahalanobis distance and so on. The query collaborators are then ranked by distance, the smaller the distance is, the higher the rank. The distances of query collaborators can be normalized, for example, $normalized\_distance = \frac{distance-min}{max-min}$, in which max and min represent the largest and smallest distance. The normalized distances can then be converted to weights, $normalized\_weight = 1 - normalized\_distance$.

Approach 2: Rank by PageRank of query collaborators.

PageRank is one of the most popular ranking algorithms proposed in [71], and was designed to compute the ranks of web pages in the whole world wide web by leveraging the hyperlinks in the web pages. The intuition is that a web page that is linked by many high ranked web pages should also receive a high rank.

Formally, let $G = (V, E)$ be a directed graph in which V is the set of vertices and E is the set of edges. For a given vertex v_i, let $in(v_i)$ be the set of vertices that link to it, and let $Out(v_i)$ be the set of vertices that vertex v_i links to. The ranking score for the PageRank is computed as $PageRank(v_i) = \frac{1-d}{N} + d \times \sum_{(v_j,v_i)\in E} \frac{PageRank(v_j)}{\text{Outdegree}(v_j)}$

where N is the number of pages in the web and d is called damping parameter $0 < d < 1$.

The above formula works well for unweighted and directed graph, in other words, if a query collaborator has forward links to other query collaborators or have backward links from other query collaborators, then the formula can be applied.

There is another case that the formed graph is undirected and weighted, for example, if each query collaborator represents a document, then query collaborators are connected by a weighted value that measures the relationship between the two documents. For undirected and weighted graph, the PageRank formula can be revised accordingly ([67]), i.e.,

$$PageRank(v_i) = \frac{1-d}{N} + d \times \sum_{(v_j,v_i)\in E} \frac{w_{ji}}{\sum_{(v_j,v_k)\in E} w_{jk}} PageRank(v_j)$$

This modified version of PageRank has been shown to work well for such NLP problems as document summarization [67] and word sense disambiguation [83].

In the above modified formula, the weight can be computed through different ways varied by applications. For example, in our case study of entity linking, the weight can be computed as the document similarity of two query collaborators.

8.2.3 Enhancing MaCR

In section 3.2, we discussed several ways to compute the combined strengths of "ranker collaborators", however, the rankers do not communicate (or interact) with each other and therefore, each ranker does not have chance to improve itself. This is counter-intuitive since in human collaborative learning, for example, students can gain new knowledge and improve their skills after interacting with teachers or other students.

Wei et. al. [91] proposed an interactive algorithm *iRank* in which two rankers can teach each other so as to jointly improve the ranking performance of both rankers. More specifically,

one base ranker takes the most confident ranking results from the other ranker as feedback to update its own ranking results, and vise visa. This process continues iteratively until a termination condition is satisfied (e.g., the ranking results do not change any more). However, the *iRank* algorithm also has limitations:

(1) The *iRank* does not explicitly indicate that the two rankers can be re-learned during the iterations because in their algorithm description, we do not see that their rankers are feeded with training data or the parameters in the rankers can be updated. From the case study (query-focused summarization) of *iRank*, we noticed that they implemented two unsupervised rankers, and only in the first iteration, the two rankers produce initial ranking scores. In the later iterations, the ranking score of each object from one ranker is updated depending on how similar it is with the top N ($N = 1$) ranked objects from the other ranker.

(2) The *iRank* does not tell how the rankers get updated if the number of rankers is greater than 2.

A possible direction to extend the *iRank* algorithm is to take into accounts of unlabeled data and multiple rankers.

It is known that it is expensive to obtain training data for supervised ranking, for example, in information retrieval, for a given query and retrieved list of documents, the annotators need to provide a relevance judgment for each document, e.g., a class label such as irrelevant or relevant; an integer number such as k which indicates the ranking position in the list; a similarity score which indicates the relevant of the document. However, it is relative easier to obtain a lot of unlabeled data. Similar with classification problems, we can design a general algorithm (Algorithm 12) which can use a small set of training data and a relative large set of unlabeled data. We then can study whether the multiple rankers can improve themselves by leveraging large amount of unlabeled data.

In Algorithm 12, the training set contains N pairs of (query, object list), each query q corresponds to a list of objects $o^{(q)} = \left\{o_1^{(q)}, \ldots, o_{n^{(q)}}^{(q)}\right\}$ that need to be ranked. $y^{(q_i)} = \left\{y_1^{(q_i)}, \ldots, y_{n^{(q_i)}}^{(q_i)}\right\}$ corresponds to the gold ranking labels for the list of objects. Let $x_j^{(q_i)} = \varphi(q_i, o_j^{(q_i)})$ denote a feature vector associated with each query-object pair $(q_i, o_j^{(q_i)})$.

We also have a set of unlabeled data which consist of M pairs of (query, object list), how-

ever, we do not know the gold ranking labels for the object list. We call each (query, object list) as an *instance* in the algorithm.

Algorithm 12 Boosting algorithm for ranking

Input:

a set of training data T:

$$\{\left(x_j^{(q_i)}, y_j^{(q_i)}\right)\}_{j=1,...,n^{(q_i)};i=1,...,N}$$

a set of unlabeled data U:

$$\{x_j^{(q_i)}\}_{j=1,...,n^{(q_i)};i=1,...,M}$$

a set of m ranking functions $f_1, f_2, ..., f_m$, some of which are supervised.

Initialization:

Create a pool U' of instances by removing P random instances from U.

repeat
  for $i = 1 \rightarrow m$ **do**
    if f_i is a supervised ranker **then**
      train it on T, and label the data in U';
      select the most confidently labeled K instances and add them to T (without duplicates)
    end if
  end for
  Refill U' with instances from U, and keep the size of U' as constant P
until termination condition is satisfied (e.g., iteration counter reaches a preset number)

In Algorithm 12, the supervised rankers can get updated (i.e., the parameters in the supervised model can be dynamically updated) whenever there are new training data available.

There are two main issues that can differentiate the above algorithm from the general boosting algorithm for classification:

(1) In ranking, different supervised methods utilize an instance in different ways, i.e., for pointwise method, an instance is split into multiple query-object pairs which are then used as training samples; for pairwise method, multiple tuples of (query-object-object) are made from an instance and are then used as training samples; only for listwise method, an instance is also a training sample.

(2) In ranking, a confidently labeled instance has different meanings according to which supervised method is applied , i.e., for pointwise method, an instance is confidently labeled if there are many highly confidently labeled (query-object) pairs made from the instance; for pairwise method, an instance is confidently labeled if there are many highly confidently labeled (query-object-object) tuples made from the instance; for listwise method, an instance is con-

fidently labeled if there are many high scores among the (query-object) pairs in the instance. To compute the confidence of an instance, we intend to use summation of confidence values of labeled samples (query-object, query-object-object etc.).

In Algorithm 12, each ranker contributes its top K confidently labeled instances, and then they are added into training set without duplicates. An alternative way to get more confidently labeled instances is to pick out those frequently occurred instances that are confidently labeled by as many rankers as possible. We can use simple voting method to find such frequently occurred instances.

The above proposed algorithm needs to be validated in some real applications, such as entity linking.

8.2.4 Enhancing MiCC

In Chapter 5, we have proposed a simple strategy to select collaborative instances by randomly selection from a relatively larger pool. This strategy is successful for document clustering, but is only partially successfully (works for some baseline clustering algorithms) for name entity clustering. We expect the following algorithm may lead to better result.

The intuition of this selection strategy by ranks of collaborative instances is that the high ranked collaborative instances can provide more information to help identify the good structure of the original set of instances in a better and faster way.

8.2.5 A Third Application of Collaborative Clustering: Event Coreference Resolution

The task of within-document event coreference resolution is defined in the Automatic Content Extraction (ACE) program, i.e., organize the event mentions in a document into equivalent classes so that all the mentions in a given class refer to a unified event.

In our previous work, we

(1) modeled the problem as a clustering problem, and applied two clustering algorithms, one is spectral graph clustering algorithm [14] and the other is agglomerative clustering algorithm[1]

[1]this agglomerative clustering algorithm is different from any of 9 nine agglomerative clustering algorithms

Algorithm 13 MiCC algorithm by ranking selection.

Input:

$X = \{x_1, x_2, ..., x_n\}$: instances in the data set;
$Y = \{y_1, y_2, ..., y_m\}$: a pool of candidate collaborative instances;
$\mathcal{F}$: a clustering algorithm;
$\mathcal{R}$: a graph ranking algorithm;
$\mathcal{M}$: an internal measure;//Assume the measure takes maximum as optimal
n_{step}: maximum number of collaborative instances picked in each iteration;
$n_{iterations}$: number of iterations

Output:

a set of best collaborative instances: $BestC$;
the optimal value computed by the internal measure LM_{opt};

1: Apply $\mathcal{F}$ on X and output a clustering, compute the clustering quality by the internal measure $\mathcal{M}$.
2: Rank collaborative instances (from high to low) in Y by the graph ranking algorithm R.
3: **for** $j = 1 \rightarrow n_{iterations}$ **do**
4: initialize a list of best found candidate collaborative clustering instances LC
5: initialize a list of best found values of clustering quality LM
6: Pick n_{step} collaborative clustering instances (naming the set as Y_i) from Y and produce a new set of instances by $X \cup Y_i$.
7: Apply $\mathcal{F}$ on $X \cup Y_i$, compute clustering quality LM_i using $\mathcal{M}$.
8: **end for**
9: Find LM_u such that $LM_u >= LM_i$ for $i \in \{1, ..., n_{iterations}\}$
10: $LM_{opt} = LM_u$
11: $BestC = LC_1 \cup ... \cup LC_u$
12: **return** $BestC$ and LM_{opt}

[19];

(2) explored two ways of computing coreference likelihood of two event mentions, one is to compute a formula combined from overlapping statistics of triggers and event arguments, and the other is to compute a supervised probability model which incorporates rich syntax, semantic features [14];

(3) studied feature contributions of supervised model in computing coreference likelihood [19].

Event coreference resolution is an even harder problem than name entity clustering. From the aspect of complexity of an instance, we can see that an instance in document clustering is simply a document, in name entity clustering, it consists of a query name and its context document, while in event coreference resolution, an instance is an event mention which contains

discussed in Chapter 5

more complex structures including an event trigger which indicates the event type and several event arguments which indicate the roles participated in the event. We can extend the previous work from the following aspects:

(1) MiCC.

A motivating example why MiCC can work is shown in figure 8.1:

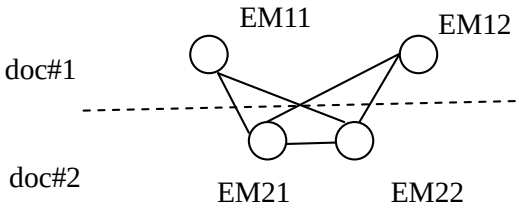


Figure 8.1: A motivating example of searching collaborative instances.

Document #1:

EM11: Yesterday in Los Angeles, pin-up icon Bettie Page succumbed to complications from a heart attack suffered almost three weeks ago.

EM12: The details of her death are known as follows.

Document #2:

EM21: Bettie Page, the 1950s pin-up model who helped set the stage for the 1960s sexual revolution, has died.

EM22: Her agent, Mark Roesler, said Page died yesterday at a Los Angeles hospital after suffering a heart attack on December 2.

In the above example, EM11 and EM12 in document #1 are two event mentions that need to be clustered, and the two mentions are coreferential because both indicate the same "DIE" event of "Bettie Page". Unfortunately, due to the data scarcity in the training corpus, the word "succumbed" in EM11 fails to be identified with the meaning of "death", and therefore, the coreference score between EM11 and EM12 may be low. The problem can be alleviate if we have a topically related document #2 in which there are two event mentions EM21 and EM22. Now all the four mentions can be grouped into a strong cluster because EM11 is strongly coreferential with EM21 and EM22 due to the overlapping arguments (e.g., Bettie Page, page, Los Angeles, heart attack), and EM12 is also strongly coreferential with EM21 and EM22 because "die" and "death" are often witnessed to be coreferential in the training corpus.

To retrieve topically related documents from ACE 2005 corpus, we can apply the topic

model with biased propagation (TMBP) algorithm described in [29, 58] which incorporates both the textual information (bag of words) and semantic information in multiple objects in a heterogeneous information network (in our case, objects include entities such as persons and organizations). The underlying intuition is that the topic of a document not only depends on the inherent content of the text, but also correlates with the entities in the documents. The basic idea of TMBP is to regularize a statistical topic model along with a biased regularization on the heterogeneous information network such that the biased regularization can exploit valuable and reinforced information from heterogeneous network. The experiments in [29] show that TBMP can significantly work better than other state-of-the-art topic modeling approaches. Hao et al. [58] also show that the topically related documents extracted by TMBP can significantly help the task of event extraction.

(2) MaCC

In our previous work, we only tried very limited clustering algorithms. We can simply extend the algorithms discussed in this thesis to this problem, and meanwhile, we can also study whether the combination schemes are still effective in this application.

References

[1] E. Amigo, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 2008. xii, 69, 70, 71, 76

[2] J. Artiles, J. Gonzalo, and S. Sekine. Weps 2 evaluation campaign: overview of the web people search clustering task. *WePS 2009*, 2009. 102

[3] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. *Proc. COLING 98*, 1998. 72

[4] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Technical Report 638, Statistics Department, University of California, Berkeley*, 2003. 15

[5] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *Proceedings of COLT: Proceedings of the Workshop on Computational Learning Theory*, 1998. 1

[6] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. *Proceedings of the 22th International Conference on Machine Learning (ICML 2005)*, 2005. 15

[7] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Comm. in Statistics,vol. 3, no. 1, pp. 1C27*, 1974. 44

[8] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise

approach to listwise approach. *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 129–136, 2007. 15, 28, 31

[9] T. Cassidy, Z. Chen, J. Artiles, H. Ji, H. Deng, L. A. Ratinov, J. Zheng, J. Han, and D. Roth. Cuny-uiuc-sri tac-kbp2011 entity linking system description. *Proc. Text Analytics Conference (TAC2011)*, 2011. 12

[10] E. Charniak and M. Johnson. Coarseto-fine-grained n-best parsing and discriminative reranking. *ACL-05*, pages 173–180, 2005. 2

[11] W. Chen, T.-Y. Liu, Y. Lan, Z. Ma, and H. Li. Ranking measures and loss functions in learning to rank. *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 315–323, 2009. 15

[12] Y Chen and J. Martin. Towards robust unsupervised personal name disambiguation. *EMNLP 2007*, 2007. 102, 110, 112

[13] Z. Chen. Graph-based clustering and its application in coreference resolution. *Technical Report TR-2010006, GC, CUNY*, 2010. xii, 12, 18, 50, 69, 71

[14] Z. Chen and H. Ji. Graph-based event coreference resolution. *Proc. ACL-IJCNLP 2009 workshop on TextGraphs-4: Graph-based Methods for Natural Language Processing*, 2009. 12, 148, 149

[15] Z. Chen and H. Ji. Language specific issue and feature exploration in chinese event extraction. *Proc. HLT-NAACL 2009*, 2009. 12

[16] Z. Chen and H. Ji. Can one language bootstrap the other: A case study on event extraction. *Proc. HLT-NAACL 2009 Workshop on Semi-supervised Learning for Natural Language Processing*, 2009. 12

[17] Z. Chen and H. Ji. Graph-based clustering for computational linguistics: A survey. *Proc. ACL 2010 TextGraphs5*, 2010. 12, 18

[18] Z. Chen and H. Ji. Collaborative ranking: A case study on entity linking. *EMNLP*, 2011. 12

[19] Z. Chen, H. Ji, and R. Haralick. A pairwise coreference model, feature impact and evaluation for event coreference resolution. *Proc. RANLP 2009 workshop on Events in Emerging Text Types*, 2009. 12, 149

[20] Z. Chen, S. Tamang, A. Lee, X. Li, W.-P. Lin, M. Snover, J. Artiles, M. Passantino, and H. Ji. Cunyblender tac-kbp2010 entity linking and slot filling system description. *Proceedings of Text Analytics Conference (TAC2010)*, 2010. 12, 27, 28

[21] Z. Chen, S. Tamang, A. Lee, X. Li, and H. Ji. A toolkit for knowledge base population. *SIGIR*, 2011. 28

[22] P. Christen. A comparison of personal name matching: Techniques and practical issues. *in Workshop on Mining Complex Data(MCD06),IEEE ICDM06*, 2006. 103

[23] W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. *Proc. IJCAI-03 Workshop on Information Integration ont the Web*, 2003. 113

[24] M. Collins. Discriminative reranking for natural language parsing. *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 175–182, 2000. 2

[25] T. M. Cover and J. A. Thomas. *Elements of Information Theory, Second Edition*. Wiley-Interscience, 2006. 72

[26] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. *EMNLP2007*, 2007. 26

[27] D. Davies and D. Bouldin. A cluster separation measure. *IEEE PAMI, vol. 1, no. 2, pp. 224C227*, 1979. 46

[28] M. DeGroot and M. Schervish. Probability and statistics (3rd edition). *Addison Wesley*, 2001. 75

[29] H. Deng, J. Han, B. Zhao, Y. Yu, and C. X. Lin. Probabilistic topic models with biased propagation on heterogeneous information network. *Proc. KDD2011*, 2011. 151

[30] B. Dom. An information-theoretic external cluster-validity measure. *IBM Research Report*, 2001. 69

[31] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. Entity disambiguation for knowledge base population. *Proceedings of COLING 2010*, 2010. 26, 28

[32] J. C. Dunn. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics, 4 (1974), pp. 95C104*, 1974. 45

[33] X. Z. Fern and C. E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. *Proceedings of the Twenty First International Conference on Machine Learning*, 2004. 59, 63

[34] B. Fischer and J. M. Buhmann. Bagging for path-based clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(11):1411C1415*, 2003. 58

[35] A. Fred. Finding consistent clusters in data partitions. *In Multiple Classifier Systems, volume LNCS 2096, pages 309C318. Springer*, 2001. 58

[36] A. Fred and A. K. Jain. Data clustering using evidence accumulation. *In Proc. of Sixteenth International Conference on Pattern Recognition, pages IV:276C280*, 2002. 59

[37] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research, 4:933-969*, 2003. 15

[38] R. Ghaemi, N. Sulaiman, H. Ibrahim, and N. Mustapha. A survey: Clustering ensembles techniques. *World Academy Science, Engineering and Technology, 38*, 2009. 1

[39] M. Halkidi and M. Vazirgiannis. Clustering validity assessment: Finding the optimal partitioning of a data set. *ICDM, pp. 187C194*, 2001. 48

[40] M. Halkidi, M. Vazirgiannis, and Y. Batistakis. Quality scheme assessment in the clustering process. *PKDD, pp. 265C276*, 2000. 47, 48

[41] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Clustering validity checking methods: Part ii. *ACM SIGMOD Record, 31(3):19C27*, 2002. 42, 50

[42] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations, 11(1)*, 2009. 116

[43] D. J. Han. Klue annotation guidelines - version 2.0. *Technical Report RC25042, IBM Research*, 2010. 105, 110

[44] X. Han and L Sun. A generative entity-mention model for linking entities with knowledge base. *Proceedings of ACL-HLT 2011*, 2011. 27

[45] S. Hoi and R. Jin. Semi-supervised ensemble ranking. *Proceedings of the 23rd AAAI Conf. on Artificial Intelligence*, 2008. 16

[46] L. Huang. Forest reranking: Discriminative parsing with non-local features. *Proceedings of ACL-HLT-08*, pages 586–594, 2008. 2

[47] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification,vol. 2, no. 1, pp. 193C218*, 1985. 49

[48] A. K. Jain, M. N. Murty, and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys,31(3):264-323*, 1999. 18

[49] H. Ji, R. Grishman, Z. Chen, and P. Gupta. Cross-document event extraction and tracking: Task, evaluation, techniques and challenges. *Proc. RANLP 2009*, 2009. 12

[50] H. Ji, R. Grishman, H. T. Dang, and K. Griffit. An overview of the tac2010 knowledge base population track. *Proceedings of Text Analytics Conference (TAC2010)*, 2010. 5, 26, 104

[51] T. Joachims. Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press*, 1999. 28

[52] T. Joachims. Optimizing search engines using clickthrough data. *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD 2002)*, 2002. 15

[53] T. Joachims. Training linear svms in linear time. *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2006. 28

[54] G. Karypis. Cluto ł software for clustering high-dimensional datasets. *version 2.1.1*, 2007. 76

[55] M.G. Kendall. *Rank Correlation Methods*. New York: Hafner Publishing Co., 1955. 75

[56] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of ACM (JASM)*, 1999. 14

[57] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly,2(83)*, 1955. 62

[58] H. Li, H. Ji, H. Deng, and J. Han. Exploiting background information networks to enhance bilingual event extraction through topic modeling. *Proc. International Conference on Advances in Information Mining and Management (IMMM2011)*, 2011. 151

[59] P. Li, C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. *Advances in Neural Information Processing Systems 20 (NIPS2007)*, 2007. 15

[60] Y. Lin. Support vector machines and the bayes rule in classification. *Data Mining and Knowledge Discovery*, pages 259–275, 2002. 15

[61] Y. Liu, Z. Li, H. Xiong, X. Gao, and J Wu. Understanding of internal clustering validation measures. *Proceedings of the 2010 IEEE International Conference on Data Mining, p.911-916*, 2010. 42, 50

[62] H. Luo, F. Jing, and X. Xie. Combining multiple clusterings using information theory based genetic algorithm. *IEEE International Conference on Computational Intelligence and Security, vol. 1, pp. 84-89*, 2006. 58

[63] C. D. Manning, P. Raghavan, and H. Schütze. Introduction to information retrieval. *Cambridge University Press*, 2008. 16, 29

[64] U. Maulik and S. Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE PAMI, vol. 24, pp. 1650C1654*, 2002. 42, 49

[65] P. McNamee and H. Dang. Overview of the tac 2009 knowledge base population track. *Proceedings of TAC*, 2009. 26

[66] M. Meila. Comparing clusterings. *In Proceedings of COLT03*, 2003. 69

[67] R. Mihalcea and P. Tarau. Textrank - bringing order into texts. *Proceedings of EMNLP 2004*, 2004. 145

[68] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika, 50 (1985), pp. 159C179*, 1985. 50, 84

[69] R. Nallapati. Discriminative models for information retrieval. *Proceedings of SIGIR*, 2004. 15

[70] F. J. Och. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. PhD thesis, Computer Science Department, RWTH Aachen, Germany, 2002. 2

[71] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Technical report, Stanford Digital Library Technologies Project*, 1998. 14, 144

[72] M. F. Porter. An algorithm for suffix stripping. *Program,14:3, 130C137*, 1980. 75

[73] T. Qin, X.-D. Zhang, M.-F. Tsai, D.-S. Wang, T.-Y. Liu, and H. Li. Query-level loss functions for information retrieval. *Information Processing and Management*, 2007. 15

[74] D. Ravichandran, E. Hovy, and F. J. Och. Statistical qa - classifier vs. re-ranker: What's the difference? *Proceedings of the ACL Workshop on Multilingual Summarization and Question Answering*, 2003. 2

[75] S. E. Robertson, S. Walker, S. Jones, M. H. Beaulieu, and M. Gatford. Okapi at trec-3. *Proceedings of the Third Text REtrieval Conference (TREC 1994)*, 1994. 14

[76] L. Rokach. Ensemble-based classifiers. *Artif Intell Rev DOI 10.1007/s10462-009-9124-7*, 2009. 1, 16

[77] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. *In Joint Conference on Empirical Methods in Natural Language Processing and Computational Language Learning*, 2007. 70, 73

[78] P. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math., 20, 1987, pp. 53-65*, 1987. 45

[79] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM, vol. 18, nr. 11, pp. 613-620*, 1975. 66

[80] S. Sharma. Applied multivariate techniques. *John Wiley & Sons, Inc*, 1996. 49

[81] L. Shen, A. Sarkar, and F. J. Och. Discriminative reranking for machine translation. *Proceedings of HLT-NAACL*, 2005. 2

[82] J. Shi and J. Malik. Normalized cuts and image segmentation. *Machine Intelligence*, 22 (8):888–905, 2000. 29, 44

[83] R. Sinha and R. Mihalcea. Unsupervised graph-based word sense disambiguation using measures of word semantic similarity. *Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2007)*, 2007. 145

[84] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *KDD*, 2000. vii, 56, 57, 65

[85] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research, 3: 583-617*, 2002. 1, 59, 60, 62, 63

[86] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu. Rankclus: Integrating clustering with ranking for heterogeneous information network analysis. *Proc. 2009 Int. Conf. on Extending Data Base Technology (EDBT'09)*, 2009. 143

[87] P. N Tan, M. Steinbach, and V. Kumar. Introduction to data mining. *Addison Wesley*, 2005. 41

[88] A. Topchy, A. K. Jain, and W. Punch. Combining multiple weak clusterings. *Proceeding of the Third IEEE International Conference on Data Mining*, 2003. 58

[89] A. Topchy, A. Jain, and W. Punch. A mixture model for clustering ensembles. *In Proc. SIAM Data Mining, pages 379C390*, 2004. 59, 61

[90] L. Vendramin, R.J.G.B. Campello, and E.R. Hruschka. On the comparison of relative clustering validity criteria. *Sparks*, 2009. 42, 51, 84

[91] F. Wei, W. Li, and S. Liu. irank: A rank-learn-combine framework for unsupervised ensemble ranking. *Journal of the American Society for Information Science and Technology,61: 1232C1243. doi: 10.1002/asi.21296*, 2010. 1, 17, 145

[92] J. Wu, H. Xiong, and J. Chen. Adapting the right measures for k-means clustering. *SIGKDD09,pages 877C886*, 2009. 66, 70, 74, 76

[93] X. L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE PAMI, vol. 13, no. 8, pp. 841C847*, 1991. 49

[94] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Trans. Neural Networks,16, pp. 645-678*, 2005. 18

[95] X. Yang, J. Su, and C. L. Tan. A twin-candidate model for learning-based anaphora resolution. *Computational Linguistics*, 34(3):327–356, 2008. 2

[96] M. Yoshida, M. Ikeda, S. Ono, I. Sato, and H. Nakagawa. Person name disambiguation by boostrapping. *SIGIR*, 2010. 28

[97] T. Zhang. Statistical analysis of some multicategory large margin classification methods. *Journal of Machine Learning Research, 5, 1225-1251*, 2004. 15

[98] W. Zhang, J. Su, C. L. Tan, and W.T. Wang. Entity linking leveraging automatically generated annotation. *Proceedings of COLING 2010*, 2010. 27

[99] Y. Zhao and G. Karypis. Comparison of agglomerative and partitional document clustering algorithms. *Technical report , University of Minnesota*, 2002. x, 80, 81, 82

[100] Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning, 55(3):311C331*, 2004. x, 42, 65, 80, 81, 82

[101] Y. Zhao and G. Karypis. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery, Vol. 10, No. 2, pp. 141 - 168*, 2005. x, 80, 81, 82

[102] Z. Zheng, F. Li, M. Huang, and X. Zhu. Learning to link entities with knowledge base. *Proceedings of HLT-NAACL*, 2010. 27, 28, 31