2011

CLUSTER-BASED TERM WEIGHTING AND DOCUMENT RANKING MODELS

Keerthiram Murugesan

University of Kentucky, keerthi166@gmail.com

CLUSTER-BASED TERM WEIGHTING AND DOCUMENT RANKING MODELS

THESIS

A thesis submitted in partial
fulfillment of the requirements for
the degree of Master of Science in
the College of Engineering at the
University of Kentucky

By
Keerthiram Murugesan
Lexington, Kentucky

Director: Dr. Jun Zhang, Professor of Computer Science
Lexington, Kentucky 2011

ABSTRACT OF THESIS

CLUSTER-BASED TERM WEIGHTING AND DOCUMENT RANKING MODELS

A term weighting scheme measures the importance of a term in a collection. A document ranking model uses these term weights to find the rank or score of a document in a collection. We present a series of cluster-based term weighting and document ranking models based on the $TF-IDF$ and *Okapi* $BM25$ models. These term weighting and document ranking models update the inter-cluster and intra-cluster frequency components based on the generated clusters. These inter-cluster and intra-cluster frequency components are used for weighting the importance of a term in addition to the term and document frequency components. In this thesis, we will show how these models outperform the $TF-IDF$ and *Okapi* $BM25$ models in document clustering and ranking.

KEYWORDS: Term weighting scheme; Document Clustering; Information Retrieval; Page Ranking; Data Mining

Author's signature: Keerthiram Murugesan

Date: September 29, 2011

CLUSTER-BASED TERM WEIGHTING AND DOCUMENT RANKING MODELS

By
Keerthiram Murugesan

Director of Thesis: Jun Zhang

Director of Graduate Studies: Raphael Finkel

Date: September 29, 2011

RULES FOR THE USE OF THESIS

Unpublished thesis submitted for the Master's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the thesis in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this thesis for use by its patrons is expected to secure the signature of each user.

Name	Date

THESIS

Keerthiram Murugesan

The Graduate School
University of Kentucky
2011

Dedicated to my parents.

ACKNOWLEDGMENTS

I would like to show my gratitude to several people who in one way or another contributed and involved in the preparation and completion of this research work.

First and foremost, I offer my utmost gratitude to my adviser, Dr. Jun Zhang, Director of Laboratory for High Performance Scientific Computing and Computer Simulation (HiPSCCS lab) for giving his continuous feedback, guidance and support. His motivation helped me to finish this research work on time.

I thank my committee members, Dr. Mukesh Singhal and Dr. Tingjian Ge for their invaluable time and suggestions.

I am indebted to my fellow researchers at HiPSCCS Research lab for reviewing my research ideas and providing valuable suggestions. Especially, Lian Liu and Xiwei Wang who helped me to get started on my thesis research.

I would like to thank Dr. Raphael A. Finkel for helping me to edit this document. It is a pleasure to thank Dr. Gary G. Schroeder and Dr. Jerzy W. Jaromczyk who stood by my side and provided their assistance in form of advice and support.

I take this opportunity to thank the High Performance Computing team, University of Kentucky for providing an access to the university supercomputer. I also thank the Graduate School, University of Kentucky for awarding Student Support Funding to attend the research conferences that are crucial for this thesis work.

Finally, I thank my parents, family and friends for their encouragement and inspiration.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Algorithms

Chapter 1 Introduction

The Vector Space Model (VSM) represents a document using a vector of T unique terms in a collection (T-dimension). Each term in a vector is associated with a weight (term weight) [CK99, SBMM96]. The term weight is based on the frequency with which the term appears in that document. The term weighting scheme measures the importance of a term with respect to a document and a collection. A term with higher weight is more important than a term with lower weight. Each document can be located in T-dimensional space, where T is the number of unique terms in a collection (Euclidean space).

With a document represented by a location in Euclidean space, we can compare any two documents by measuring the actual distance between them. In the same way, a user-supplied query can be represented as a vector and mapped in Euclidean space. In order to find a set of documents relevant to a query, we can find documents that are closer to this query in Euclidean space. A document ranking model finds the similarities between these documents and a query. If a document is more relevant to a query, it will get a higher ranking.

VSM and term weighting schemes are widely used in many research areas such as document clustering, classification, information retrieval, document ranking, etc.

The main contributions of this thesis are discussed in the next two sections.

1.1 Term Weighting Scheme

Unlike the traditional approaches, the term weighting schemes, discussed in this thesis (CBT), use the generated clusters to update the weight of each term. These term weighting schemes for document clustering are based on the traditional $TF - IDF$.

Our motivation is based on the idea that the terms within a cluster have more similarity than the terms distributed across different clusters. We concentrated on the term distribution within a cluster to measure the term weights. We implemented this idea by giving more weight to the terms that are common within a cluster but uncommon in other clusters. Our experiment shows that the proposed term weighting schemes based on clusters give better results than the other well-known term weighting schemes traditionally used for document clustering.

1.2 Cluster-Based Retrieval

Cluster-based retrieval uses the cluster hypothesis to retrieve a set of documents relevant to a query [LC04].

Cluster hypothesis *Documents in the same cluster behave similarly with respect to relevance to information needs* [Voo85].

If a document in a cluster is relevant to a query, then the rest of the documents in that cluster are potentially relevant to the same query.

There are two approaches in cluster-based retrieval. The first approach retrieves one or more clusters relevant to a query instead of retrieving documents relevant to a query. In other words, this approach retrieves and ranks the relevant clusters instead of the relevant documents. Based on the cluster hypothesis, the documents from the highly ranked clusters are more relevant to a query than the documents from the clusters with lower ranking. The main motive of this approach is to achieve higher efficiency and faster search.

The second approach uses the generated clusters as a reference in improving the retrieved relevant documents. In this approach, the given document collection is clustered (static clustering) beforehand. When a set of documents is retrieved for a query, the generated clusters (static clusters) of the collection are used as a reference to update the retrieved relevant document list. The main goal of this approach is to improve the precision-oriented searches.

This thesis presents a method similar to the second approach. It performs the clustering on the given collection in advance. It uses these static clusters to update document ranking. As explained in the previous section (CBT), the proposed cluster-based retrieval is based on an idea that terms that are unique to a static cluster are more important than the terms that are repeated in most of the static clusters.

This unsupervised method identifies the important terms in a cluster using their frequencies within a static cluster and compares them with the frequency of these terms outside this static cluster. It integrates the information gathered from these static clusters into the document ranking model.

1.3 Thesis Overview

This thesis presents new approaches in assigning weight to each term in a document. Prior works, discussed in Chapter 2, modified the existing term weighting scheme to improve the efficiency of an algorithm.

Our models use the generated clusters for weighting the importance of a term in a document. The experimental results in this thesis are based on the application of our term weighting schemes in document clustering (Chapters 3, 4 and 6) and ranking (Chapters 5 and 6).

Chapter 3 presents the basic form of our term weighting scheme, Cluster-Based Term weighting scheme (CBT), for improving the quality of the clusters generated by a document clustering algorithm (partitional or hierarchical). The proposed term weighting scheme uses the information gathered from the generated clusters as separate components (inter- and intra- cluster frequency components) in addition to the term frequency and document frequency components. On each iteration, CBT updates the weight of all unique terms of each cluster in a collection.

CBT penalizes terms of a document that are misplaced in a cluster. In other words, important terms of documents that are in a right cluster will get a higher weight with respect to their frequencies, whereas terms of documents that are in a wrong cluster will be penalized and get a lower weight.

Chapter 4 introduces a variant of CBT ($CBTV$) that increases the weight of important terms in a document (with respect to their frequency within a cluster),

without penalizing the terms of a document that are misplaced in a cluster. This $CBTV$ is used for cluster-based retrieval application. Chapter 5 presents the usage of $CBTV$ in document ranking. The experimental results compare $CBTV$ with that of the $TF - IDF$ in document ranking.

Chapter 6 introduces another variant of CBT based on *Okapi* $BM25$ model: CBT $BM25$. The chapter discusses the experimental comparison between *Okapi* and CBT $BM25$ models for document clustering and ranking. Finally, Chapter 7 concludes with the summary of this thesis.

1.4 Outline

The rest of this thesis is organized as follows: Chapter 2 describes the related works and motivations behind this work. Chapter 3 presents the basic form of CBT and compares its experimental results with that of the $TF-IDF$ term weighting scheme. Chapter 4 then presents a variant of CBT ($CBTV$) to address the penalizing problem in CBT. Chapters 3 and 4 show the application of CBT and $CBTV$ in document clustering. Chapter 5 shows the application of $CBTV$ in document ranking. Chapter 6 compares CBT $BM25$ with one of the most popular term weighting and document-ranking models, *Okapi* $BM25$. Chapter 7 concludes this thesis by summarizing the significance of this contribution.

Chapter 2 Related Works

This chapter discusses the basic knowledge about the information retrieval, document representation, document clustering and document ranking. These concepts will be repeatedly used throughout this thesis. We start with the general search engine architecture, showing how web search works. This general architecture will help us to understand the basic operations in information retrieval process. Then we discuss the document representation and term weighting schemes and their application in document clustering and ranking. Finally, we present the motivation behind this thesis.

2.1 How Web Search Works

The general search engine architecture (Figure 2.1) consists of the following main components:

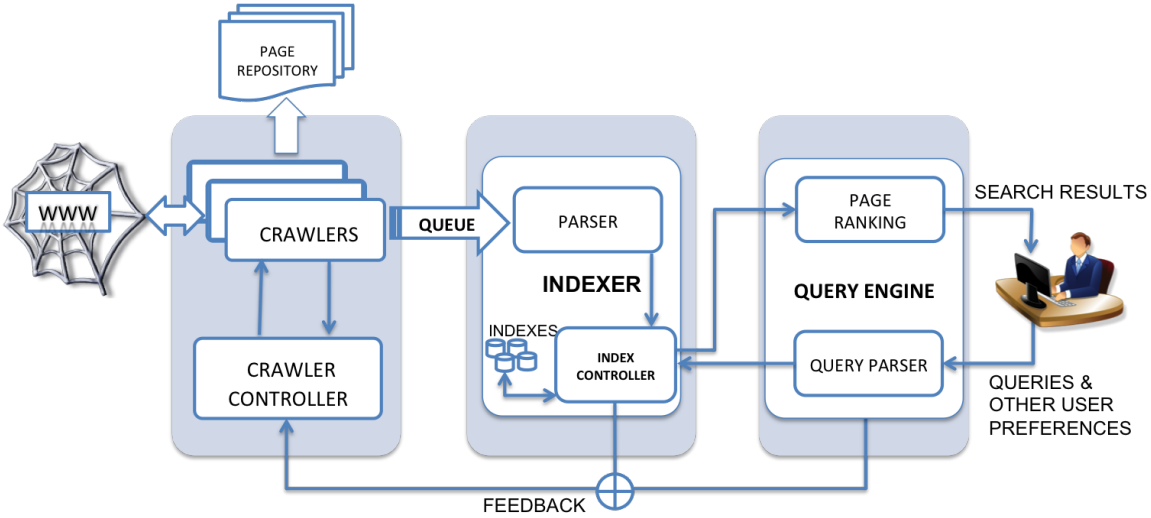

Figure 2.1: General Search Engine Architecture

Crawler

The crawler is an important component of the search engine architecture; it hops from one web document to another on the Internet in search of any useful information, based on the set of URLs provided by the crawler controller. One or more crawlers run concurrently to maximize the searching for information. When a crawler fetches a web document, it extracts all the URLs in the document and gives them to the crawler controller. The crawler controller organizes these URLs based on the feedback information received from the indexer and query engine components. The

crawler adds these web documents to the queue for further processing by the indexer component. Optionally, a copy of the web document is stored in the page repository for future use. We will not be discussing about crawlers further in this thesis.

Indexer

The indexer uses a parser to extract all the terms and constructs a vector of terms (VSM) for each web document in the queue. The representation of documents as vectors will be discussed in Section 2.2. These document vectors are stored along with their link information in the indexes using the index controller. These vectors use term weights to represent the importance of a term in a document (Section 2.3). The document vectors with the term weights are used for many applications such as document clustering (Section 2.4), document ranking (Section 2.5), etc.

Query Engine

When a user sends a query to the query engine, the engine extracts all the important terms from the query using the query parser and constructs a vector of terms. This query parser is similar in functionality to the index parser. The query vector is then sent to the index controller, which uses the indexes to generate a list of web pages that are relevant to the query using a document ranking model (Section 2.5).

Although there are many modules in the search engine architecture, we focused on these three main components to understand the basic operation of a search engine. Readers are referred to [MRS08] for the complete description of the search engine architecture.

2.2 Document Representation

Each document fetched by crawlers in Figure 2.1 needs to be stored in a search engine. Due to the space and memory constraints, a search engine cannot handle a document in its raw format. It is necessary to represent each document in a compact, easy-to-use, standard format.

The boolean retrieval model is one of the most popular information retrieval models which represents each document as a set of unique terms in a collection. Each term is mapped to [0,1]. If the $term_t$ occurs in the document d_i, then the $term_t$ will have value 1; otherwise it will have value 0.

$$d_i = \{term_1, w_{i1}; term_2, w_{i2}; \cdots term_T, w_{iT}\}. \tag{2.1}$$

$$w_{it} = \begin{cases} 1 & \text{if } term_t \text{ is in } d_i \\ 0 & \text{if } term_t \text{ is not in } d_i \end{cases}$$

Like the boolean retrieval model, the vector space model (VSM) uses a bag of words approach in which each unique term in a collection is mapped to $\mathbb{R}^+$. Each $term_t$ in the document d_i is assigned the weight w_{it} which represents its importance.

VSM is the fundamental model in the information retrieval applications such as clustering, ranking and classification. Each document in a collection is represented as a vector of unique terms. These T-dimensional document vectors can be placed in Euclidean space, in which there is one axis for each term. In this representation, the order of the terms in the vector is not important. Figure 2.2 shows a 3-dimensional vector in Euclidean space [SM86].

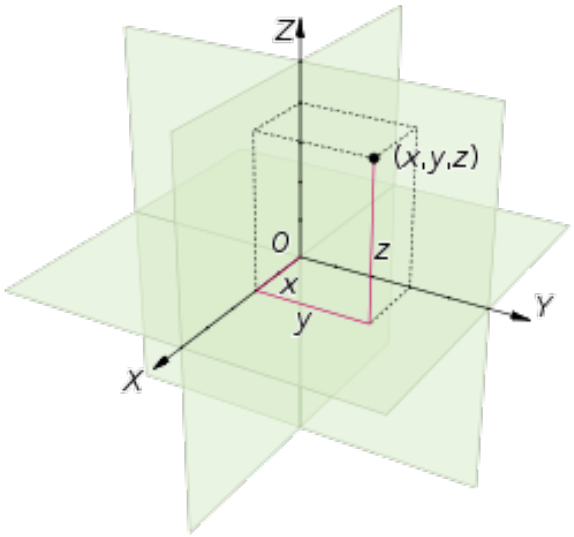

Figure 2.2: 3-Dimensional Vector Space (Euclidean Space from Wikipedia)

Euclidean Distance and Cosine Similarity Measures

In VSM, the Euclidean distance measures the distance between any two document vectors in Euclidean space [XW05], whereas the cosine similarity measures the similarity between any two document vectors in Euclidean space.

The Euclidean distance between the document vectors x and y is given as:

$$D(x,y) = \sqrt{\sum_{i:\ 1\cdots T} (x_i - y_i)^2} \tag{2.2}$$

where T is the total number of terms in the collection ζ. The cosine similarity between the document vectors x and y is given as:

$$Cos(x,y) = \frac{x.y}{\|x\|\|y\|} \tag{2.3}$$

where the $x.y$ represents the dot product of the document vectors x and y. The dot product $x.y$ of the document vectors x and y is defined as:

$$x.y = \sum_{i=1}^{T} x_i.y_i \tag{2.4}$$

$\|x\|$ and $\|y\|$ represent the Euclidean lengths of the document vectors x and y respectively. The Euclidean length in Equation 2.3 is used for normalizing the value using the document length. The Euclidean length of the document vector x is defined as:

$$\|x\| = \sqrt{\sum_{i=1}^{T} x_i^2} \tag{2.5}$$

If two documents x and y are similar, then the cosine similarity measure will be closer to 1; otherwise it will be closer to 0.

2.3 Term Weighting Scheme

As explained in the previous section, the boolean retrieval model assigns 1 or 0 based on the presence or absence of a term in a document. This model performs undesirably in querying for a document. Later, VSM was introduced for ranked retrieval [Sal89]. It is widely used in querying documents, clustering, classification and other information retrieval applications because it is simple and easy to understand. It uses a bag of word approach. Each document d_i in the collection ζ is represented as a vector of terms with weight [CO05, JPD00].

For instance, one of the most commonly used term weighting schemes, $TF-IDF$, assigns weights to each term using the term frequency (tf) and inverse document frequency (idf). The term frequency of the $term_t$ is the number of times the given $term_t$ occurs in a document. The inverse document frequency is the total number of documents in a collection containing the $term_t$ with respect to the total number of documents (N) in a collection. Then, the document vector, d_i, in Equation 2.1 is represented as:

$$d_i = \{term_1, tf_{i1}.\log\frac{N}{N_1}; term_2, tf_{i2}.\log\frac{N}{N_2}; \cdots term_T, tf_{it}.\log\frac{N}{N_t};\}. \tag{2.6}$$

The term weight w_{it} determines whether the $term_t$ will be included in the further steps. As mentioned in [SB88], only certain terms extracted from a document can be used for identifying and scoring a document within the collection. The term weighting schemes are used to assign weight to each term in a document. These term weights represent the importance of a term with respect to a collection. Document clustering uses these term weights to compare two documents for their similarity. Several term weight schemes are in use today; but none of them are specific to the clustering algorithms [JPD00].

Table 2.1 shows representation of some of the term weighting schemes commonly used. Here, TF is the Term Frequency, IDF is the Inverse Document Frequency and ICF is the Inverse Cluster Frequency.

w_{itj} is the weight of the term $term_t$ in the document d_i of the cluster C_j.

$tf_{it} = f_{it}$ is the term frequency of the term $term_t$ in the document d_i.

$idf_t = \log\frac{N}{N_t}$ is the inverse document frequency for the term $term_t$ in the collection ζ, where N is the total number of documents in the collection and N_t is the number of documents that contain the term $term_t$.

Table 2.1: Term weighting schemes.

	Term weighting schemes
$Norm - TF$	$w_{it} = \frac{f_{it}}{\sqrt{\sum_T f_{it}^2}}$
$TF - IDF$	$w_{it} = f_{it} \log \frac{N}{N_t}$
$TF - IDF - ICF$	$w_{itj} = f_{it} \log \frac{N}{N_t} \log \frac{K}{K_t}$

$icf_t = \log \frac{K}{K_t}$ is the inverse cluster frequency of the term $term_t$ in the collection ζ, where K is the total number of clusters in the collection and K_t is the number of clusters that contains the term $term_t$.

2.4 Document Clustering

A document clustering algorithm helps to find groups in documents that share a common pattern [TSK05, SKK00, GRS98, ZKF05, CKPT92]. It is an unsupervised technique and is used to automatically find clusters in a collection without any user supervision.

The main goal of the clustering is to find the meaningful groups so that the analysis of all the documents within clusters is much easier compared to viewing it as a whole collection. There are different ways to cluster documents. But, most commonly, two types of clustering methods are used: Partitional and Hierarchical clustering [MZ11].

Partitional Clustering

A partitional clustering algorithm finds all the non-overlapping clusters at once by dividing the set of documents based on an objective function [JD88, Mac67, NH94, CS96, ZHD+01, HXZ+98, Bol98, SG00, DHZ+01]. These algorithms try to minimize or maximize an objective function. Most partitional clustering algorithms are prototype-based, in which a prototype for each cluster is chosen and the documents are grouped based on these prototypes. These prototypes are called centroids. Usually these algorithms run several times until a convergence occurs or an optimum condition is met.

K-means Clustering Algorithm

The K-means clustering algorithm is a popular unsupervised partitional clustering method [MRS08, Mac02]. The K-means clustering algorithm tries to minimize the following objective function $O(N, K)$:

$$O(N, K) = \sum_{j=1}^{K} \sum_{d_i \in C_j} \|d_i - c_j\|^2 \tag{2.7}$$

where $\|d_i - c_j\|^2$ is the distance between the document d_i and the centroid c_j. The centroid of the documents in a cluster C_j can be computed as:

$$c_j = \frac{1}{|C_j|} \sum_{d_i \in C_j} d_i \tag{2.8}$$

The following algorithm shows the K-means clustering for generating K clusters on the document collection ζ.

Algorithm 1 K-means partitional clustering algorithm

Require: An integer $K \geq 1$, Document Collection ζ.

1: **if** $K = 1$ **then**
2: **return** ζ
3: **else**
4: Initialize $l = 0$
5: $\{C_1^{(0)}, \cdots, C_K^{(0)}\} \leftarrow RANDOM - CLUSTERS(\zeta, K)$
6: **repeat**
7: **for all** $d_i \in \zeta, i : 1 \ldots N$ **do**
8: $m = \arg\min_j |c_j - d_i|$
9: $C_m^{(l+1)} \leftarrow C_m^{(l+1)} \cup d_i$
10: **end for**
11: $l \leftarrow l + 1$
12: **for** $j = 1$ **to** K **do**
13: $c_j \leftarrow \frac{1}{|C_j^{(l)}|} \sum_{d_i \in C_j^{(l)}} d_i$
14: **end for**
15: **until** No change in K centroids
16: **return** $\{C_1^{(l)}, \cdots, C_K^{(l)}\}$
17: **end if**

Initially, the K-means algorithm needs a set of random cluster to start with. $RANDOM - CLUSTERS(\zeta, K)$ in the algorithm 1 generates a set of K random clusters. The runtime complexity of the K-means algorithm 1 is $O(LNK)$ where L is the total number of iterations in the outer loop, N is the total number of documents in a collection and K is the total number of clusters.

Hierarchical Clustering

Hierarchical clustering generates a tree of clusters by splitting / merging clusters on each level until the desired number of clusters are generated. This generated tree is often called *Dendogram* [GRS98, SS73, Kin67, KHK99], shown in Figure 2.3.

Hierarchical clustering can use top-down approach (Divisive) or bottom-up approach (Agglomerative) to construct the dendogram. Agglomerative clustering starts with one document in each cluster (singleton cluster) and repeatedly merges two clusters that are most similar in their pattern at each step until a single cluster

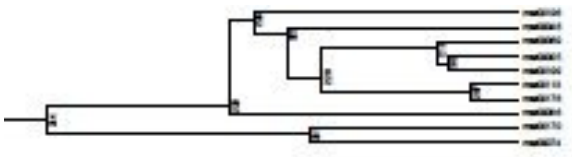

Figure 2.3: Dendogram

of all the documents is obtained. Divisive clustering, on the other hand, starts with all documents as a single cluster and splits them until all the clusters are singleton. These types of clustering are the most commonly used because their underlying representation of clusters in hierarchy resembles their application domain [ZKF05, XW05, CKPT92].

Hierarchical agglomerative clustering algorithms merge a pair of clusters at each step based on one of the following linkage metrics for measuring proximity between the clusters [GRS98, JD88, SS73, Kin67, KHK99, GRS99].

The Single Link Algorithm (SLA) measures the maximum of the pair-wise similarity of the documents from each cluster to merge a pair of clusters [SS73].

$$Similarity_{SLA}(C_A, C_B) = \max_{x \in C_A, y \in C_B} \cos(x, y) \tag{2.9}$$

The Complete Link Algorithm (CLA) measures the minimum of the pair-wise similarity of the documents from each cluster to merge a pair of clusters [Kin67].

$$Similarity_{CLA}(C_A, C_B) = \min_{x \in C_A, y \in C_B} \cos(x, y) \tag{2.10}$$

The Group Average Algorithm (unweighted pair group method with arithmetic mean (UPGMA)) measures the average of the pair-wise similarity of the documents from each cluster to merge a pair of clusters [JD88].

$$Similarity_{UPGMA}(C_A, C_B) = \frac{1}{n_i n_j} \sum_{x \in C_A, y \in C_B} \cos(x, y) \tag{2.11}$$

where $\cos(x, y)$ is the cosine similarity between two documents or clusters represented as vectors.

Though there are many other linkage metrics, Equations (2.9),(2.10) and (2.11) are commonly used. These methods use the Euclidean distance or inter-cluster similarity matrix for their comparisons and measurements. Experiments have shown that UPGMA performs better than SLA and CLA.

Divisive hierarchical clustering algorithms pick a big cluster or cluster with the lowest intra-cluster similarity measure to split on each level [ELL01, KR90]. Most recently, partitional clustering algorithms have been used to split the clusters. The bisecting K-means algorithm [SKK00] is a typical divisive hierarchical clustering algorithm that uses the K-means partitional clustering algorithm to split its cluster.

Bisecting K-means Clustering Algorithm

The bisecting K-means algorithm is a hierarchical divisive clustering algorithm. The bisecting K-means clustering bisects the largest remaining clusters into two subclusters at each iteration using the K-means partitional clustering algorithm, until the desired number of clusters (K) is obtained.

The following algorithm shows the bisecting K-means clustering for generating K clusters on the document collection ζ. In Algorithm 2, C is a set of clusters.

Algorithm 2 Bisecting K-means hierarchical divisive clustering algorithm

Require: An integer $K \geq 1$, Document Collection ζ.

1: **if** $K = 1$ **then**
2: **return** ζ
3: **else**
4: Initialize $C \leftarrow \{\zeta\}$
5: **for** $k = 1$ **to** $K - 1$ **do**
6: $L \leftarrow PICK - LARGEST - CLUSTER(C)$
7: $\{C_1, C_2\} \leftarrow K - MEANS(L, 2)$
8: $C \leftarrow C \cup \{C_1, C_2\}$
9: **end for**
10: **return** C
11: **end if**

$PICK - LARGEST - CLUSTER(C)$ removes and returns the largest cluster in the set C. The runtime complexity of the bisecting K-means algorithm 2 is linear on the number of documents, N.

Notations

We used the following notations throughout this thesis: C is the set of clusters in a collection, $C = \{C_1, C_2, \cdots C_k\}$ and Ω is the set of known classes, $\Omega = \{\Omega_1, \Omega_2 \cdots \Omega_l\}$. $C_1, C_2, \cdots C_k$ are the clusters produced by a clustering algorithm and $\Omega_1, \Omega_2 \cdots \Omega_l$ are the known classes where $l = k$ in our experiment. $|\Omega_i|$ represents the number of documents in the class Ω_i, $|C_j|$ is the number of documents in the cluster C_j, $|\Omega_i \cap C_j|$ represents the number of documents appeared in both class Ω_i and cluster C_j, N is the total number of documents in a collection and c is the number of known classes in a collection.

Evaluation Measures

Cluster validity measures the goodness of the clusters generated by the clustering algorithm. In order to compare the term weighting schemes in document clustering, we used the classification-oriented measures. In other words, the performance of a clustering algorithm can be measured using internal quality criterion and/or external quality criterion. In this paper, we used the external quality criterion, which

compares the clustering result produced by the clustering algorithm with the known classes. These classes are identified based on human judgment. We used five external quality measures in this paper: Entropy, F-Measure, Purity, Mutual Information and Normalized Mutual Information [TSK05, SKK00, ZKF05].

Entropy

The entropy provides a measure of uncertainty [Sha48]. The entropy of the cluster C_j can be defined as:

$$H_j = -\sum_{i:\,1\cdots l} p_{ij}\log(p_{ij}) \tag{2.12}$$

$$H_j = -\sum_{i:\,1\cdots l} \frac{|\Omega_i \cap C_j|}{|C_j|}\log\frac{|\Omega_i \cap C_j|}{|C_j|}, j:\,1\cdots k \tag{2.13}$$

where j is in $1\cdots k$. Equations (2.12) and (2.13) are equivalent. Equation 2.13 is the maximum likelihood estimate (MLE) of the probabilities in Equation 2.12. The total entropy of a clustering algorithm, H, is the weighted sum of the entropies of all the clusters in a collection.

$$H = \sum_{j:\,1\cdots k} \frac{H_j * |C_j|}{N} \tag{2.14}$$

F-Measure

The F-Measure provides a measure of accuracy [LA99]. It is based on recall and precision measures used for the evaluation of an information retrieval system. Precision is the fraction of a cluster that consists of documents of a specified class. Recall is the extent to which a cluster contains all the documents of a specified class [TSK05]. Precision and recall can be computed as:

$$Recall, R(\Omega_i, C_j) = \frac{|\Omega_i \cap C_j|}{|\Omega_i|} \tag{2.15}$$

$$Precision, P(\Omega_i, C_j) = \frac{|\Omega_i \cap C_j|}{|C_j|} \tag{2.16}$$

F-Measure between the class Ω_i and cluster C_j is given by:

$$F(\Omega_i, C_j) = \frac{2 * R(\Omega_i, C_j) * P(\Omega_i, C_j)}{R(\Omega_i, C_j) + P(\Omega_i, C_j)} \tag{2.17}$$

The F-Measure of any class Ω_i is the maximum F-Measure value obtained in any node of the hierarchical clustering. The total F-Measure is the weighted average of the F-Measure values of all the classes

$$\text{F-Measure}(\Omega_i) = \max F(\Omega_i, C_j), C_j \in C \tag{2.18}$$

$$\text{F-Measure} = \sum_{i:\,1\cdots c} \frac{|\Omega_i|}{N} * \text{F-Measure}(\Omega_i) \tag{2.19}$$

Purity

The purity measures the quality of the clusters. The purity of the cluster C_j is given by:

$$Purity(\Omega_i, C_j) = \max_i |\Omega_i \cap C_j| \tag{2.20}$$

$$Purity = \sum_{j:\, 1 \cdots k} \frac{Purity(\Omega_i, C_j)}{N}, \Omega_i \in \Omega \tag{2.21}$$

Mutual Information

The mutual information (I) measures the amount of information by which our knowledge about the classes increases when we are told what the clusters are.

$$I(\Omega, C) = \sum_j \sum_i P(\Omega_i \cap C_j) \log \frac{P(\Omega_i \cap C_j)}{P(\Omega_i) P(C_j)} \tag{2.22}$$

$$I(\Omega, C) = \sum_j \sum_i \frac{|\Omega_i \cap C_j|}{N} \log \frac{N|\Omega_i \cap C_j|}{|\Omega_i||C_j|} \tag{2.23}$$

where $P(\Omega_i \cap C_j)$, $P(\Omega_i)$ and $P(C_j)$ are the probabilities of a document being in cluster C_j, in class Ω_i, and in both cluster C_j and class Ω_i respectively. Equations 2.22 and 2.23 are equivalent. Equation 2.23 is the maximum likelihood estimate (MLE) of the probabilities in Equation 2.22.

Normalized Mutual Information

The Normalized Mutual Information (NMI) is a normalized variant of the mutual information to address the problem of singular clusters getting higher I value.

$$NMI(\Omega, C) = \frac{I(\Omega, C)}{[H_i + H_j]/2} \tag{2.24}$$

where $I(\Omega, C)$ is the mutual information, H_i and H_j are the entropies of the class Ω_i and cluster C_j, respectively.

2.5 Document Ranking

VSM discussed in Section 2.2 can be used to find a list of documents relevant to a query [Buc, BMWC98, LCS97, SM86, Jon81].

In VSM, documents are relevant only if they are closer to the query in Euclidean space as shown in Figure 2.2. In Figure 2.4, documents d_1 and d_2 are closer to the given query q, which makes d_1 and d_2 relevant. But it is hard to find the documents relevant to a query visually. The cosine similarity, discussed in Section 2.2, can be used to find the relevant documents. The cosine angle between the query q and the documents can show how far a document is from the query.

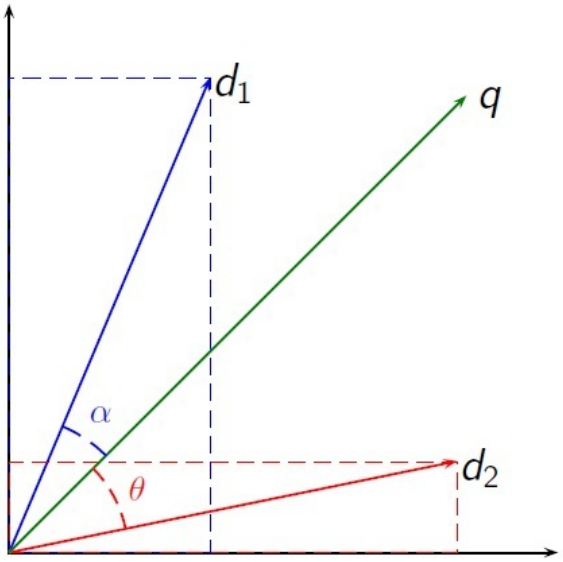

Figure 2.4: Ranking Documents in the Vector Space Model

In the above diagram, if the cosine angle between the document d_1 and query q, α, is less than the cosine angle between the document d_2 and query q, θ, then we can say, document d_1 is more relevant to the query q than the document d_2. The cosine similarity measure for ranking is repeated below for more clarity.

$$Score(q, d) = \frac{q.d}{\|q\|\|d\|} \tag{2.25}$$

In some cases, the normalization may be removed if the document and query length has no impact on the returned relevant document list. Then, the $Score(q, d)$ is simply the dot product of query and document vectors.

$$Score(q, d) = q.d \tag{2.26}$$

Using Equation 2.26, we can introduce the term weighting scheme into the ranking model. For instance, if the $TF - IDF$ is used as a term weighting scheme then, Equation 2.26 can be written as:

$$Score(q, d_i) = \sum_{t \in q} tf - idf_{t,i} \tag{2.27}$$

$$Score(q, d_i) = \sum_{t \in q} tf_{it}. \log \frac{N}{N_t} \tag{2.28}$$

where $tf - idf_{t,i}$ is the term weight of the term $term_t$ in the document d_i, tf_{it} is the term frequency of the term $term_t$ in the document d_i, N is the total number of documents in a collection and N_t is the number of documents in a collection containing the term $term_t$. Equations 2.27 and 2.28 are equivalent. With this representation, any term weighting scheme can be used to score a document for the given query q.

Document Ranking Algorithm

The algorithm for ranking/scoring documents for a query is given in Algorithm 3. This algorithm first takes one query term at a time. For each query term, it compute the partial rank/score for each document in the collection ζ. The final score/rank of each document in the collection ζ is computed when all the query terms are processed. This approach is called *breadth first search*. An alternative approach, *depth first search*, computes the final score/rank for each document, before computing a score for the next document.

Algorithm 3 Document Ranking Algorithm

Require: Query q, and Document Collection ζ.

1: $Score[N] = 0.0$
2: **for all** term $term_t$ in query q **do**
3: **for all** document d_i in collection ζ **do**
4: $w_{it} \leftarrow tf_{it}.\log \frac{N}{N_t}$
5: $Score[i] \leftarrow Score[i] + w_{it}$
6: **end for**
7: **end for**
8: $R \leftarrow SORT(\zeta, Score[])$
9: **return** R

Evaluation Measures

Several measures are available to evaluate the relevancy of the retrieved, ranked document list. Evaluation of information retrieval systems involves a notion of relevant and irrelevant documents. A document is relevant if it answers the query. It is difficult to evaluate since the relevancy is based on the user's judgment. A document that is relevant to a query for one user might be irrelevant for some other user. The set of retrieved documents is based on the top k retrieved documents.

Precision and recall are the two basic measures that use relevancy in the retrieved and non-retrieved list of documents for a query. With this definition, Equations 2.16 and 2.15 can be redefined in the information retrieval field as follows:

$$Precision, P = \frac{No.\ of\ relevant\ documents\ retrieved}{Total\ No.\ of\ documents\ retrieved} \tag{2.29}$$

$$Recall, R = \frac{No.\ of\ relevant\ documents\ retrieved}{Total\ No.\ of\ relevant\ documents} \tag{2.30}$$

The computed precision and recall measures can be used to plot the *precision-recall curve*, as shown in Figure 2.5. If the $(k+1)^{th}$ document is irrelevant, then recall is same for the top k documents, but precision has dropped. If the $(k+1)^{th}$ document is relevant, then both precision and recall are increased.

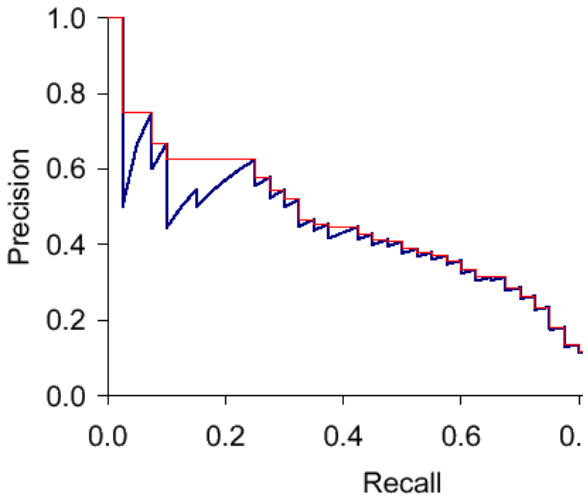

Figure 2.5: Precision-Recall Curve [MRS08]

We used three standard measures, based on precision (Equation 2.29) and recall (Equation 2.30), that are commonly used in the TREC conferences: Mean Average Precision (MAP), Precision at 10 (P@10) and Precision at 100 (P@100). The following definitions are adopted from [MRS08].

Precision at 10 (P@10)

Precision at 10 measures the fraction of relevant document in the top 10 retrieved documents.

Precision at 100 (P@100)

Precision at 100 measures the fraction of relevant document in the top 100 retrieved documents.

Mean Average Precision (MAP)

Average Precision is the average of the precision at K values computed after each relevant document is encountered. Mean Average Precision is the arithmetic mean of average precision values for individual user queries. Average Precision is simply an approximate area under the precision-recall curve. MAP is the average of area under the precision-recall curve of each query.

If the set of relevant documents (of size m_j) for a query $q_j \in Q$ is $\{d_1, d_2, \cdots d_{m_j}\}$ and R_{jk} is the set of ranked retrieval results from the top result until document d_k, then

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk}) \tag{2.31}$$

2.6 Motivations

Several term weighting schemes have been proposed to compute the importance of a term in a document and in a collection. $Norm - TF$, $TF - IDF$, ATC, LTU and *Okapi* are some of the widely used term weighting schemes [SB88, SW97]. One of the most commonly used term weighting schemes is the $TF - IDF$. It measures the importance of a term using its frequency within a document and the inverse of its document frequency within a collection. This thesis extends this idea to the document clustering and ranking algorithms.

Several papers have suggested modifying an existing term weighting scheme for their methods. [AK02] shows a modified $TF - IDF$ term weighting scheme to avoid single terms from getting a higher weight. [RJP+06] proposes a new term weighting scheme $TF - ICF$ (Term Frequency Inverse Corpus Frequency) for clustering a dynamic data stream. It uses the existing collections to weight the terms in the data stream. Also, [ZWLX09] used $DF - ICF$ with inter- and intra-cluster components for extracting a description from a cluster.

In this thesis, we extend the idea of [RJP+06] and the cluster components used in [ZWLX09] to introduce a new term weighting scheme that efficiently uses the cluster information obtained from the clustering algorithm. In addition, we discuss the variants of CBT and application of these CBT variants in document clustering and ranking.

Chapter 3 Cluster-Based Term Weighting Scheme

A term weighting scheme can be used to identify the importance of each term with respect to a collection and assigns weights to them accordingly. Several term weight schemes are in use today, but none of them are specific to the clustering algorithms.

In this chapter, we present a new term weighting method based on the traditional $TF-IDF$ term weighting scheme. Our motivation is based on the idea that the terms of the documents in a same cluster have similar importance compared to the terms of the documents in a different cluster. We concentrated on the terms that are important within a cluster and considered the other terms as irrelevant and redundant. We implemented this idea by giving more weight to the terms that are common within a cluster but uncommon in other clusters.

In order to experiment our new term weighting scheme, we used both unsupervised partitional (K-means) and hierarchical (Bisecting K-means) clustering algorithms [MRS08, Mac02, SKK00]. First, we ran the K-means algorithm with the four term weighting schemes, discussed in the previous chapter, to show that the CBT term weighting scheme improves the quality of the clusters generated by a partitional clustering algorithm. Next, we ran the bisecting K-means algorithm with the $TF-IDF$ and CBT to compare the clusters generated by these term weighting schemes. The results of these algorithms directly reflect the impact of the term weighting schemes in the clustering algorithm.

From our experiment, we found that the new term weighting scheme based on the clusters gives better results than the other well-known term weight schemes traditionally used for both partitional and hierarchical clustering algorithms.

3.1 The Proposed Term Weighting Scheme

In this section, we introduce our new term weighting scheme. For the term $term_t$, document d_i and cluster C_j, CBT is given as:

$$w_{itj} = tf_{it}.idf_t.df_{tj}.icf_t \tag{3.1}$$

$$= f_{it}.\log\frac{N}{N_t}.\frac{df_j}{|C_j|}.\log\frac{K}{K_t} \tag{3.2}$$

Equation (3.1) is equivalent to Equation (3.2). Here, $df_{tj} = \frac{df_j}{|C_j|}$ is the document frequency of the term $term_t$ within the cluster C_j, where df_j is the number of documents in the cluster C_j that contain the term $term_t$, and $|C_j|$ is the total number of documents in the cluster C_j.

Our new term weighting scheme has four components. The first two components are based on the term weighting components discussed in [SB88]. The last two components are the cluster components as shown in Table 3.1.

In other words, CBT assigns a weight to a term which is

Table 3.1: List of Components in CBT term weighting scheme.

Component	Description
tf_{it}	Term Frequency Component. High when term t occurs often in a document i.
idf_t	Collection Frequency Component. High when term t occurs less often in the entire collection.
df_{tj}	Intra-cluster Frequency Component. High when term t occurs more often in a cluster j.
icf_t	Inter-cluster Frequency Component. High when term t occurs less often in clusters other than cluster j.

- Highest when the term occurs more frequently in the documents of a cluster and uncommon in other clusters.
- Higher when the term occurs less frequently in the documents of a cluster and uncommon in other clusters.
- Lower when the term occurs often in a few clusters.
- Lowest when the term occurs in most of the documents in a collection.

The following section shows the modified K-means and bisecting K-means algorithms using CBT as a term weighting scheme for computing the importance of a term in a document.

K-means algorithm with CBT

Initially, the K-means algorithm doesn't have any information about the cluster components, so we start the algorithm by setting df_{tj} and icf_t to 1 as shown in line 4 and update the inter- and intra-cluster components on each iteration. If a document has a set of terms that doesn't belong to a cluster, then its term weight will be reduced so that it will move to other clusters. It will be repeated until it finds a suitable cluster of its type.

The runtime complexity of the traditional K-means algorithm is $O(LNK)$ where L is the total number of iterations in the outer loop, N is the total number of documents

Algorithm 4 K-means partitional clustering algorithm with CBT

Require: An integer $K \geq 1$, Document Collection ζ.

if $K = 1$ **then**
  return ζ
else
  Initialize $l = 0$
  $\{C_1^{(0)}, \cdots, C_K^{(0)}\} \leftarrow RANDOMCLUSTERS(\zeta, K)$
  repeat
    for all $d_i \in \zeta, i\colon 1 \ldots N$ **do**
      $m = \arg\min_j |c_j - d_i|$
      $C_m^{(l+1)} \leftarrow C_m^{(l+1)} \cup d_i$
    end for
    $l \leftarrow l + 1$
    $w_{itj} \leftarrow tf_{it}.idf_t.df_{tj}.icf_t$; for each term $term_t$ in a document d_i of a cluster $C_j^{(t)}$, $t\colon 1 \ldots T$, $i\colon 1 \ldots N$, $j\colon 1 \ldots K$
    for $j = 1$ **to** K **do**
      $c_j \leftarrow \frac{1}{|C_j^{(l)}|} \sum_{d_i \in C_j^{(l)}} d_i$
    end for
  until No change in K centroids
  return $\{C_1^{(l)}, \cdots, C_K^{(l)}\}$
end if

in a collection and K is the total number of clusters. Algorithm 4 updates the term weights for all clusters on each iteration to reflect the changes made in the new clusters, so it takes $O(LNK + LK) = O(LNK)$ since $LK < LNK$.

Bisecting K-means algorithm with CBT

Like the K-means algorithm in the previous section, df_{tj} and icf_t are set to 1 initially.

The runtime complexity of the bisecting K-means algorithm is linear with the number of documents in the collection ζ. Algorithm 5 updates the term weights for all clusters for K times to reflect the changes made in the new clusters, so it takes $O(N + K^2) = O(N)$ since $K << N$.

3.2 Data Collections Used For CBT

We used the TREC [TRE99], 20 Newsgroup [Lan] and Reuters-21578 [Lew99] data collections for our experiment. TR11, TR12, TR23, TR31, and TR45 collections are taken from TREC-5, TREC-6 and TREC-7. 20 NG S1 - S5 are the five randomly chosen subsets of 20 Newsgroup documents [ZZH07]. RE S1 and RE S2 data sets are from Reuters-21578 collection. For the RE S1 data set, we filtered documents from the original Reuters-21578 data set that belongs only to a single category. We

Algorithm 5 Bisecting K-means hierarchical divisive clustering algorithm with the Cluster-Based Term weighting scheme

Require: An integer $K \geq 1$, Document Collection ζ.

1: **if** $K = 1$ **then**
2: **return** ζ
3: **else**
4: Initialize $C \leftarrow \{\zeta\}$
5: **for** $k = 1$ **to** $K - 1$ **do**
6: $L \leftarrow PICK - LARGEST - CLUSTER(C)$
7: $\{C_1, C_2\} \leftarrow K - MEANS(L, 2)$
8: $C \leftarrow C \cup \{C_1, C_2\}$
9: $w_{itj} \leftarrow tf_{it}.idf_t.df_{tj}.icf_t$; for each term $term_t$ in a document d_i of a cluster $C_j^{(t)}$ in C, t: $1 \ldots T$, i: $1 \ldots N$, j: $1 \ldots K$
10: **end for**
11: **return** C
12: **end if**

got 4645 documents that have only one category. In addition to that, we used the Reuters transcribed subset (RE S2) [HB99]. For all the data sets shown in Table 3.3, we removed the stop words and stemmed using the Porter stemming algorithm [Por80].

Table 3.2: Data sets used for the K-means algorithm.

Data set	Collection	# of Doc	# of Class
TR11	TREC	414	9
TR12	TREC	313	8
TR23	TREC	204	6
TR31	TREC	927	7
TR45	TREC	690	10
20 NG S1	20 Newsgroup	2000	20
20 NG S2	20 Newsgroup	2000	20
20 NG S3	20 Newsgroup	2000	20
20 NG S4	20 Newsgroup	2000	20
20 NG S5	20 Newsgroup	2000	20
RE S1	Reuters-21578	4645	59
RE S2	Reuters-21578	200	10

3.3 Experimental Results

In our first experiment, we used the K-means clustering algorithm with $Norm - TF$, $TF - IDF$, CBT and $TF - IDF - ICF$ term weighting schemes to identify the

Table 3.3: Data sets used for the Bisecting K-means algorithm.

Data set	Collection	# of Doc	# of Class
TR11	TREC	414	9
TR12	TREC	313	8
TR23	TREC	204	6
TR31	TREC	927	7
TR45	TREC	690	10
RE S1	Reuters-21578	4645	59
RE S2	Reuters-21578	200	10
20 NG	20 Newsgroup	20000	20

Table 3.4: K-means clustering algorithm - Avg. Entropy measured for $Norm - TF$, CBT, $TF - IDF - ICF$ and $TF - IDF$ term weighting schemes.

Data sets	Term weighting schemes			
	$Norm - TF$	$TF - IDF$	CBT	$TF - IDF - ICF$
TR11	0.8413	0.7905	0.8535	**0.7749**
TR12	0.9009	0.6834	**0.6139**	0.6261
TR23	1.0424	0.9246	**0.8390**	0.8501
TR31	**0.9379**	1.2781	0.9657	1.0822
TR45	1.0787	1.3469	**1.0443**	1.1485
20 NG S1	2.4824	0.4037	**0.2995**	0.4475
20 NG S2	2.4954	0.5791	**0.4164**	0.8010
20 NG S3	2.4727	0.9366	**0.5082**	0.7886
20 NG S4	2.4923	0.3138	**0.2845**	0.5210
20 NG S5	2.7464	**0.2983**	0.3274	0.5665
RE S1	**2.0103**	2.0638	**2.0116**	2.0600
RE S2	1.7281	1.7369	1.6810	**1.6711**

importance of using inter- and intra- cluster components in the term weights using the average entropy measure. Since the K-means clustering algorithm is unstable and sensitive to initial centroids, we ran the algorithm 10 times with different random seed for the initial centroids on each run. We repeated this experiment for the four term weighting schemes on the data collections listed in Table 3.3.

In addition, we used the bisecting K-means algorithm to show that the cluster-based term weighting scheme can be applied for hierarchical clustering algorithms. In this experiment, we compared the $TF - IDF$ and CBT term weighting schemes. We used the average entropy, average F-Measure and average purity measures to compare the results. Like the K-means algorithm, we ran the bisecting K-means algorithm 10 times to avoid any fluctuations in the final results.

Table 3.5: Bisecting K-means clustering algorithm - Avg. Entropy, Avg. F-Measure and Avg. Purity measured for the $TF - IDF$ and CBT term weighting schemes.

Data Source	*CBT*			TF-IDF		
	Avg. Entropy	Avg. F-Measure	Avg. Purity	Avg. Entropy	Avg. F-Measure	Avg. Purity
TR11	**1.3435**	0.2067	**0.5039**	1.4102	**0.2478**	0.4850
TR12	**1.5480**	**0.2256**	**0.3936**	1.7344	0.1946	0.3514
TR23	**1.3337**	**0.1803**	**0.4853**	1.3351	0.1719	**0.4853**
TR31	**1.2539**	**0.1473**	**0.5150**	1.4105	0.1407	0.4344
TR45	**1.5070**	**0.3003**	**0.4404**	1.5922	0.2627	0.4210
RE S1	**2.0039**	0.0504	0.4140	2.0061	**0.0519**	**0.4137**
RE S2	**1.9169**	**0.2663**	**0.2764**	1.9981	0.2444	0.2518
20 NG	2.8548	0.09863	0.1079	**2.2575**	**0.1894**	**0.2141**

We calculated the entropy for the term weighting schemes, as given in Equation (2.13), for each run after the algorithm converged. Then, we computed the average of the entropies obtained in each run. Similarly, we computed the average F-measure and average purity measures. Table 3.4 shows the average entropy calculated for each data set. Table 3.5 shows the average entropy, average F-Measure and average purity measured for the $TF - IDF$ and CBT term weighting schemes for the bisecting K-means clustering algorithm.

Both experiments show that the results obtained from the K-means and bisecting K-means clustering algorithms with the CBT term weighting scheme have shown better results compared to the other term weighting schemes on each data set.

According to the cluster-based term weighting scheme, a term is considered important to a cluster if it is unique to that cluster and occurs frequently within the documents of that cluster. The inter- and intra-cluster components try to identify these important terms by analyzing the term frequency distribution at three levels: document, cluster and collection. And our experimental results have shown that adding these cluster components in the term weighting scheme significantly improves the results on each data set. We believe that some of the deviations in the results are due to the clustering algorithms' lack of handling the noise in the data collection. The better results in each data set are boldfaced.

Chapter 4 Variant of CBT

The cluster-based term weighting scheme discussed in the previous chapter penalizes the terms of the documents that are placed in a wrong cluster. The terms of documents that are in right cluster get higher term weights. A new variant of CBT ($CBTV$) increases the weight of the important terms of the documents that are in right clusters without penalizing other terms.

$CBTV$ will be useful for document ranking where the important terms in a document should have higher term weights, which in turn improves the overall document ranking.

Like CBT in Chapter 3, we compared the $TF-IDF$ and $CBTV$. First, we ran the K-means partitional clustering algorithm with the $TF-IDF$ and $CBTV$. Then, we ran the bisecting K-means hierarchical divisive clustering algorithm with the $TF-IDF$ and CBT variant to show that $CBTV$ outperforms the traditional $TF-IDF$. The results of these algorithms directly reflect the impact of the term weighting schemes in the clustering algorithms.

This chapter concentrates on introducing a new cluster-based term weighting scheme that can be used in the upcoming chapter. The main focus of this experiment is to find whether this new term weighting scheme, similar to CBT, can be applied for both document clustering and ranking.

From our experiment, we found that $CBTV$ based on the clusters gives better results than the well-known term weight scheme ($TF-IDF$) traditionally used for both partitional and hierarchical clustering algorithms.

4.1 The Proposed Term Weighting Scheme

In this section, we introduce $CBTV$ term weighting scheme. For the term $term_t$, document d_i, and cluster C_j, $CBTV$ is given as:

$$w_{itj} = (1-\lambda)\left(\frac{tf_{it}.idf_t}{D_L}\right) + (\lambda)\left(\frac{df_{tj}.icf_t}{Cl_S}\right) \quad (4.1)$$

$$= (1-\lambda)\left(\frac{f_{it}.\log\frac{N}{N_t}}{D_L}\right) + (\lambda)\left(\frac{df_{tj}.\log\frac{K}{K_t}}{Cl_S}\right) \quad (4.2)$$

where D_L and Cl_S are the average document length and average cluster size respectively [SBMM96]. Equation (4.1) is equivalent to Equation (4.2). Here, df_{tj} is the document frequency of the term $term_t$ within the cluster C_j.

This $CBTV$ is a linear interpolation of the document and cluster models controlled by a smoothing parameter λ as shown in Table 4.1.

From Equation 4.1, one can get the normalized $TF-IDF$ term weighting scheme by taking the value of the smoothing parameter, λ as 0. The document model captures the frequency distribution of terms within a document and a collection, whereas the cluster model captures the frequency distribution of the term (using document

Table 4.1: Notations used in $CBTV$.

Notation	Description
$tf_{it}.idf_t$	Document Model. represents frequency of the term $term_t$ with respect to the document d_i and collection ζ.
$df_{tj}.icf_t$	Cluster Model. represents frequency of the document containing term $term_t$ with respect to the cluster C_j and collection ζ
λ	Smoothing Parameter. To control the influence of above two models. 4.3
D_L and Cl_S	Normalization Factors. to remove the impact of document lengths and cluster size.

frequency) within a cluster and a collection. The cluster model assigns weight to the unseen terms, for which the document model assign zero value since frequency of the term $term_t$ in document d_i, $tf_{it} = 0$. We can call this cluster model *fallback model*.

The following section shows the modified K-means and bisecting K-means algorithms using $CBTV$ as a term weighting scheme for computing the importance of a term in a document.

K-means algorithm with $CBTV$

Initially, the K-means algorithm doesn't have any information about the cluster components, so we start the algorithm by setting df_{tj} and icf_t to 1 and update the document and cluster models on each iteration.

The runtime complexity of the traditional K-means algorithm is $O(LNK)$ where L is the total number of iterations in the outer loop, N is the total number of documents in a collection and K is the total number of clusters. Algorithm 6 updates the term weights for all clusters on each iteration to reflect the changes made in the new clusters, so it takes $O(LNK + LK) = O(LNK)$ since $LK < LNK$.

Bisecting K-means algorithm with $CBTV$

Like the K-means algorithm in the previous section, df_{tj} and icf_t are set to 1 initially.

The runtime complexity of the bisecting K-means algorithm is linear with the number of documents in the collection ζ. Algorithm 7 updates the term weights for

Algorithm 6 K-means partitional clustering algorithm with $CBTV$

Require: An integer $K \geq 1$, Document Collection ζ.

if $K = 1$ **then**
  return ζ
else
  Initialize $l = 0$
  $\{C_1^{(0)}, \cdots, C_K^{(0)}\} \leftarrow RANDOMCLUSTERS(\zeta, K)$
  repeat
    for all $d_i \in \zeta, i\colon 1 \ldots N$ **do**
      $m = \arg\min_j |c_j - d_i|$
      $C_m^{(l+1)} \leftarrow C_m^{(l+1)} \cup d_i$
    end for
    $l \leftarrow l + 1$
    $w_{itj} \leftarrow (1 - \lambda)(\frac{tf_{it}.idf_t}{D_L}) + (\lambda)(\frac{df_{tj}.icf_t}{Cl_S})$; for each term $term_t$ in a document d_i of a cluster $C_j^{(t)}$, $t\colon 1 \ldots T$, $i\colon 1 \ldots N$, $j\colon 1 \ldots K$
    for $j = 1$ **to** K **do**
      $c_j \leftarrow \frac{1}{|C_j^{(l)}|} \sum_{d_i \in C_j^{(l)}} d_i$
    end for
  until No change in K centroids
  return $\{C_1^{(l)}, \cdots, C_K^{(l)}\}$
end if

Algorithm 7 Bisecting K-means hierarchical divisive clustering algorithm with $CBTV$

Require: An integer $K \geq 1$, Document Collection ζ.

if $K = 1$ **then**
  return ζ
else
  Initialize $C \leftarrow \{\zeta\}$
  for $k = 1$ **to** $K - 1$ **do**
    $L \leftarrow PICK - LARGEST - CLUSTER(C)$
    $\{C_1, C_2\} \leftarrow K - MEANS(L, 2)$
    $C \leftarrow C \cup \{C_1, C_2\}$
    $w_{itj} \leftarrow (1 - \lambda)(\frac{tf_{it}.idf_t}{D_L}) + (\lambda)(\frac{df_{tj}.icf_t}{Cl_S})$; for each term $term_t$ in a document d_i of a cluster $C_j^{(t)}$ in C, $t\colon 1 \ldots T$, $i\colon 1 \ldots N$, $j\colon 1 \ldots K$
  end for
  return C
end if

all the clusters for K times to reflect the changes made in the new clusters, so it takes $O(N + K^2) = O(N)$ since $K << N$.

4.2 Data Collections Used For $CBTV$

For this experiment on $CBTV$ term weighting scheme, we used the TREC [TRE99], WebACE [Lan] and Reuters-21578 [Lew99] data collections. FBIS, HITECH, LA1, and LA2 collections are taken from TREC-5, TREC-6 and TREC-7. RE0 and RE1 data sets are from Reuters-21578 collection. For all the data sets shown in Table 4.2, we removed the stop words and stemmed using the Porter stemming algorithm [Por80].

Table 4.2: Data sets used for both the K-means and Bisecting K-means algorithms (with $CBTV$).

Data set	Collection	# of Doc	# of Class
FBIS	TREC	2463	17
HITECH	TREC	2301	6
K1A	WEBACE	2340	20
LA1	TREC	3204	6
LA2	TREC	3075	6
RE0	REUTERS-21578	1504	13
RE1	REUTERS-21578	1657	25
WAP	WEBACE	2000	20

4.3 Parameter Selection

The document and cluster models use a smoothing parameter to control the influence of each model. The smoothing parameter (λ) takes different forms [ZL04, LC04].

- The **Jelinek-Mercer Method** is a simple mixture model where λ is an arbitrary weight between 0 and 1.
- For the **Bayesian Smoothing with the Dirichlet prior**, λ takes the following representation.

$$\lambda = \frac{\sum_{term_t \in D} tf_{it}}{\sum_{term_t \in D} tf_{it} + \mu} \tag{4.3}$$

  where $\sum_{term_t \in D} tf_{it}$ is the summation of frequencies of all the terms in the document d_i and μ is the Dirichlet smoothing coefficient which usually takes 1000.

In this experiment, we used the Bayesian Smoothing with the Dirichlet prior as our smoothing parameter. The value of the parameter depends on the collection being used for the experiment.

4.4 Experimental Results

We used the K-means and bisecting K-means clustering algorithms to compare the $TF - IDF$ and $CBTV$ term weighting schemes. Since we changed only the term weighting schemes used in the K-means and bisecting K-means algorithms, the results observed from the clusters generated by these algorithms directly represent the impact of this new variant.

For these experiments, we used the average entropy, average F-Measure, average purity, mutual information and normalized mutual information evaluation measures for both the K-means and bisecting K-means clustering algorithms to compare the $TF - IDF$ and $CBTV$. We used the algorithms from partitional and hierarchical clustering to demonstrate that $CBTV$ performs well in both clustering types.

In our first experiment with the K-means clustering algorithm, we ran the algorithm for 10 times with different random seed for the initial centroids on each run and on each data set, since the K-means algorithm is unstable and sensitive to the chosen initial centroids. We repeated the same experiment for the $TF - IDF$ term weighting scheme. We calculated the average entropy, average F-Measure, average purity, mutual information and normalized mutual information for each data set, by computing the average of the 10 values of the entropy, F-Measure, Purity, mutual information and normalized mutual information obtained from 10 runs.

Table 4.3 shows the average entropy, average F-Measure, average purity, mutual information and normalized mutual information computed for the K-means clustering algorithm with the $TF - IDF$ and CBT term weighting schemes.

We repeated the same experiment with the bisecting K-means algorithm. We used the same evaluation measures as for the K-means algorithm (average entropy, average F-Measure, average purity, mutual information and normalized mutual information). As with the K-means algorithm, we repeated the experiment 10 times with different seed for the initial centroids.

Table 4.4 shows the average entropy, average F-Measure, average purity, mutual information and normalized mutual information computed for the bisecting K-means clustering algorithm with the $TF - IDF$ and CBT term weighting schemes.

Table 4.3: K-means clustering algorithm - Avg. Entropy, Avg. F-Measure, Avg. Purity, MI and NMI measured for the $TF-IDF$ and $CBTV$ term weighting schemes.

Data Source	$CBTV$					$TF-IDF$				
	Avg. Entropy	Avg. F-Measure	Avg. Purity	MI	NMI	Avg. Entropy	Avg. F-Measure	Avg. Purity	MI	NMI
FBIS	1.1489	0.3756	**0.6117**	**1.2977**	**0.4580**	**1.1469**	**0.3814**	0.6020	1.2938	0.4567
HITECH	**1.1590**	**0.4515**	**0.5412**	**0.5129**	**0.2863**	1.2174	0.4214	0.5082	0.4562	0.2546
K1A	**1.3802**	0.3737	**0.5544**	**1.1766**	**0.3928**	1.3833	**0.3791**	**0.5542**	1.1685	0.3901
LA1	0.9176	0.6180	0.6761	0.7741	0.4320	**0.9050**	**0.6345**	**0.6915**	**0.7869**	**0.4392**
LA2	**0.8755**	**0.6365**	**0.6936**	**0.8062**	**0.4499**	0.8893	0.6326	0.6870	0.7930	0.4426
RE0	**1.0949**	**0.3125**	**0.6079**	**0.7297**	**0.2845**	1.1121	0.3021	0.5973	0.7120	0.2776
RE1	**1.1881**	**0.4266**	**0.6408**	**1.4716**	**0.4572**	1.1907	**0.4266**	0.6386	1.4690	0.4564
WAP	1.3472	**0.3751**	**0.5706**	**1.2204**	**0.4074**	**1.3465**	0.3732	**0.5706**	1.2169	0.4062

Table 4.4: Bisecting K-means clustering algorithm - Avg. Entropy, Avg. F-Measure, Avg. Purity, MI and NMI measured for the $TF - IDF$ and $CBTV$ term weighting schemes.

Data Source	$CBTV$					$TF - IDF$				
	Avg. Entropy	Avg. F-Measure	Avg. Purity	MI	NMI	Avg. Entropy	Avg. F-Measure	Avg. Purity	MI	NMI
FBIS	**1.1463**	**0.4293**	**0.6209**	**1.2687**	**0.4478**	1.1816	0.4249	0.6109	1.2298	0.4341
HITECH	**1.1263**	**0.4983**	**0.5632**	**0.5469**	**0.3052**	1.1633	0.4685	0.5345	0.5104	0.2848
K1A	**1.2138**	**0.4532**	**0.6105**	**1.3080**	**0.4366**	1.2397	0.4067	0.5959	1.2800	0.4273
LA1	0.9064	0.5836	0.6875	0.7861	0.4387	**0.8795**	**0.6201**	**0.6950**	**0.8132**	**0.4538**
LA2	**0.8730**	**0.6160**	**0.6893**	**0.8098**	**0.4520**	0.9291	0.6021	0.6607	0.7532	0.4203
RE0	**0.9848**	**0.3969**	**0.6610**	**0.8297**	**0.3235**	1.0124	0.3864	0.6493	0.8027	0.3130
RE1	**1.2124**	**0.4085**	**0.6383**	**1.4264**	**0.4431**	1.2466	0.3805	0.6333	1.3940	0.4331
WAP	**1.2412**	**0.4128**	**0.6008**	**1.2806**	**0.4275**	1.2707	0.3978	0.5931	1.2562	0.4193

From Tables 4.3 and 4.4, we observed that $CBTV$ performs better than the $TF-IDF$ in both the K-means and bisecting K-means clustering algorithms. Both these experiments have shown that using the document and cluster models in a term weighting scheme give better results than the traditional $TF-IDF$. In addition, the results for $CBTV$ are similar to the results observed for CBT discussed in Chapter 3. The better results in each data set are bold faced.

We will use $CBTV$ as a document ranking model in the next chapter.

Chapter 5 CBT for Document Ranking

A document ranking model finds the similarities between any document in a collection and a query. A document ranking algorithm retrieves a set of documents that are most relevant to a query. As explained in Section 2.5, we find the similarity/score between a document and a query using:

$$Score(q, d) = q.d \tag{5.1}$$

5.1 The Proposed Ranking Model

Any term weighting scheme can be used as a document ranking model. Section 2.5 has shown how we can use the $TF - IDF$ term weighting scheme as a document ranking model. Similarly, we can use $CBTV$ discussed in Chapter 4 as a document ranking model.

$$Score(q, d_i) = \sum_{t \in q} CBTV_{t,i} \tag{5.2}$$

$$Score(q, d_i) = \sum_{t \in q} (1 - \lambda)(\frac{tf_{it}.idf_t}{D_L}) + (\lambda)(\frac{df_{tj}.icf_t}{Cl_S}) \tag{5.3}$$

where D_L and Cl_S are the average document length and average cluster size respectively. Equation (5.2) is equivalent to Equation (5.3). Here, j in Equation (5.3) is the index of the cluster C_j of the document d_i, df_{tj} is the document frequency of the term $term_t$ within the cluster C_j.

$CBTV$ Document Ranking Algorithm

Document ranking algorithm computes the similarity between each document in the collection ζ and the given query. Based on these similarity scores / ranks computed, the retrieved list of documents potentially relevant to the query is returned as a search result. Algorithm 8 uses *breadth first search*. First, the algorithm computes a partial score/rank for each document in a collection. The final score/rank will be computed when all the query terms are handled. This algorithm needs a set of clusters (static clusters) as an input for computing the score/rank for each document. Line 4 uses these cluster information for the cluster model and computes the score using $CBTV$.

Higher the score for a document, more relevant the document is to the given query. The $SORT(\zeta, Score[])$ sorts the retrieved list based on the scores computed for each document. The ranked, retrieved list is returned as a search result for the given query.

5.2 Data Collections Used For $CBTV$ Document Ranking Model

For this experiment, we used three data sets: Genomic, AP89, and Classic3. AP89 is the collection of Associated Press (AP) news stories from 1989. Classic3 data set is

Algorithm 8 Document Ranking Algorithm using $CBTV$

Require: Query q, Cluster set $\{C_1, C_2, \cdots, C_K\}$, and Document Collection ζ.

1: $Score[N] = 0.0$
2: **for all** term $term_t$ in query q **do**
3: **for all** document d_i of cluster C_j in collection ζ **do**
4: $w_{itj} \leftarrow (1-\lambda)(\frac{tf_{it}.idf_t}{D_L}) + (\lambda)(\frac{df_{tj}.icf_t}{Cl_S})$
5: $Score[i] \leftarrow Score[i] + w_{itj}$
6: **end for**
7: **end for**
8: $R \leftarrow SORT(\zeta, Score[])$
9: **return** R

made from three data sets: MED, CRAN and CISI. Each data set contains a query set to evaluate the document ranking model.

Table 5.1: Data sets used for $CBTV$ Document Ranking Model.

Data set	# of Doc	Query Set
Genomic	42,251	1 - 50
AP89	84,640	1 - 50
Classic3 (MED)	3893	1 - 30
Classic3 (CRAN)	3893	1 - 50
Classic3 (CISI)	3893	1 - 50

Since the most common words in a data set do not contribute much to the overall score/rank, we removed these stop words from both data set and query set. Figure 5.1 shows the top 50 most frequent terms in AP89 data set. Most of the terms in the list will have no impact on the retrieved list. The terms are then stemmed using the Porter stemming algorithm [Por80].

5.3 Experimental Results

In this experiment, each data set in Table 5.1 is clustered in advance. The generated clusters, *static clusters*, will be used in the cluster model for computing the score/rank for each document in the collection ζ. These static clusters can be generated using any document clustering algorithm. In our experiment, we used the traditional K-means partitional clustering algorithm. And we used $K = \sqrt{N}$ as the number of clusters for the document clustering algorithm. We observed that the results obtained depends on the quality of the static clusters generated by a clustering algorithm.

We used MAP, P@10 and P@100 measures to evaluate the ranked list of documents for the query sets. In addition, we have shown the percentage improvement achieved through $CBTV$ compared to the document ranking model with the $TF - IDF$ term weighting scheme.

Word	*Freq.*	*r*	P_r(%)	$r.P_r$	*Word*	*Freq*	*r*	P_r(%)	$r.P_r$
the	2,420,778	1	6.49	0.065	has	136,007	26	0.37	0.095
of	1,045,733	2	2.80	0.056	are	130,322	27	0.35	0.094
to	968,882	3	2.60	0.078	not	127,493	28	0.34	0.096
a	892,429	4	2.39	0.096	who	116,364	29	0.31	0.090
and	865,644	5	2.32	0.120	they	111,024	30	0.30	0.089
in	847,825	6	2.27	0.140	its	111,021	31	0.30	0.092
said	504,593	7	1.35	0.095	had	103,943	32	0.28	0.089
for	363,865	8	0.98	0.078	will	102,949	33	0.28	0.091
that	347,072	9	0.93	0.084	would	99,503	34	0.27	0.091
was	293,027	10	0.79	0.079	about	92,983	35	0.25	0.087
on	291,947	11	0.78	0.086	i	92,005	36	0.25	0.089
he	250,919	12	0.67	0.081	been	88,786	37	0.24	0.088
is	245,843	13	0.65	0.086	this	87,286	38	0.23	0.089
with	223,846	14	0.60	0.084	their	84,638	39	0.23	0.089
at	210,064	15	0.56	0.085	new	83,449	40	0.22	0.090
by	209,586	16	0.56	0.090	or	81,796	41	0.22	0.090
it	195,621	17	0.52	0.089	which	80,385	42	0.22	0.091
from	189,451	18	0.51	0.091	we	80,245	43	0.22	0.093
as	181,714	19	0.49	0.093	more	76,388	44	0.21	0.090
be	157,300	20	0.42	0.084	after	75,165	45	0.20	0.091
were	153,913	21	0.41	0.087	us	72,045	46	0.19	0.089
an	152,576	22	0.41	0.090	percent	71,956	47	0.19	0.091
have	149,749	23	0.40	0.092	up	71,082	48	0.19	0.092
his	142,285	24	0.38	0.092	one	70,266	49	0.19	0.092
but	140,880	25	0.38	0.094	people	68,988	50	0.19	0.093

Figure 5.1: Top 50 terms of AP89. P_r is the probability of the occurrence of a term. [CMS09]

The following three tables show MAP, P@10 and P@100 measured for each data set with the given query set.

Table 5.2: MAP evaluation measure for Document Ranking Model.

Data set	# of Clusters ($K = \sqrt{N}$)	MAP		Improvement (%)
		$CBTV$	$TF - IDF$	
Genomic	206	0.2510	0.1605	+56.39
AP89	291	0.1521	0.1292	+17.73
Classic3 (MED)	62	0.5180	0.4207	+23.13
Classic3 (CRAN)	62	0.3057	0.2493	+22.62
Classic3 (CISI)	62	0.1525	0.1357	+12.38

Table 5.3: P@10 evaluation measure for Document Ranking Model.

Data set	# of Clusters ($K = \sqrt{N}$)	P@10		Improvement (%)
		$CBTV$	$TF - IDF$	
Genomic	206	0.2740	0.2740	-
AP89	291	0.1702	0.1681	+1.25
Classic3 (MED)	62	0.6433	0.5433	+18.41
Classic3 (CRAN)	62	0.2380	0.2040	+16.67
Classic3 (CISI)	62	0.2711	0.2600	+4.27

Table 5.4: P@100 evaluation measure for Document Ranking Model.

Data set	# of Clusters ($K = \sqrt{N}$)	P@100		Improvement (%)
		$CBTV$	$TF - IDF$	
Genomic	206	0.2370	0.2004	+18.26
AP89	291	0.1121	0.0964	+16.29
Classic3 (MED)	62	0.1823	0.1603	+13.72
Classic3 (CRAN)	62	0.0514	0.0486	+5.76
Classic3 (CISI)	62	0.1427	0.1456	-1.99

All the evaluation measures have shown that using $CBTV$ for document ranking has improved the final results. In some cases, we have achieved 64% improvement over traditional $TF - IDF$. The results can be expected to improve, if we use hierarchical clustering for generating the static clusters.

*Okapi BM*25 is one of the best-known term weighting and document ranking functions [RWBW99, RZRZ]. It is a probabilistic model of information retrieval. In this chapter, we will show how another variant of *CBT* performs better than *Okapi BM*25 model. *Okapi BM*25 function can be defined as:

$$w_{it} = \frac{(k_1+1)tf_{it}}{tf_{it}+k_1((1-b)+b(\frac{D_l}{D_L})}.idf_t \tag{6.1}$$

where k_1, b are tuning parameters for scaling the term frequency tf_{it} and document length respectively, D_l and D_L are the document length and average document length respectively, and df_t is the document frequency of the term $term_t$ in the collection ζ. Experiments have shown that for the values $b = 0.75$ and k_1 between 1.2 and 2, *Okapi BM*25 gives the maximum performance.

As discussed earlier, any term weighting scheme can be used for document ranking. The document ranking model for *Okapi BM*25 is given as:

$$Score(q,d_i) = \sum_{t\in q}\frac{(k_1+1)tf_{it}}{tf_{it}+k_1((1-b)+b(\frac{D_l}{D_L})}.idf_t \tag{6.2}$$

6.1 The Proposed Term Weighting Scheme

In this section, we propose a new term weighting scheme based on *CBTV*, discussed in Chapter 4, and *Okapi BM*25.

$$\begin{aligned} w_{itj} = \Bigg(& (1-\lambda)\frac{(k_1+1)tf_{it}}{tf_{it}+k_1((1-b_1)+b_1(\frac{D_l}{D_L})}.\log\frac{N}{N_t} \\ & +(\lambda)\frac{(k_2+1)df_{jt}}{df_{jt}+k_2((1-b_2)+b_2(\frac{C_s}{C_S})}.\log\frac{K}{K_t}\Bigg) \end{aligned} \tag{6.3}$$

where b_2, k_2 are similar to the tuning parameters in Equation 6.1 and $b = b_1$, C_s and C_S are the cluster size and average cluster size respectively, df_{tj} is the document frequency of the term $term_t$ in the cluster C_j.

In Chapter 5, we have shown *CBTV* based document ranking. Similarly, we can use *CBT BM*25 for document ranking.

$$\begin{aligned} Score(q,d_i) = \Bigg(& \sum_{t\in q}(1-\lambda)\frac{(k_1+1)tf_{it}}{tf_{it}+k_1((1-b_1)+b_1(\frac{D_l}{D_L})}.\log\frac{N}{N_t} \\ & +(\lambda)\frac{(k_2+1)df_{jt}}{df_{jt}+k_2((1-b_2)+b_2(\frac{C_s}{C_S})}.\log\frac{K}{K_t}\Bigg) \end{aligned} \tag{6.4}$$

The notations used in Equation 6.4 are given in Table 6.1.

Table 6.1: Notations in CBT BM25.

Notation	Description
$\frac{(k_1+1)tf_{it}}{tf_{it}+k_1((1-b_1)+b_1(\frac{D_L}{D_L}))}.\log\frac{N}{N_t}$	Document Model. represents frequency of the term $term_t$ with respect to the document d_i and collection ζ.
$\frac{(k_2+1)df_{jt}}{df_{jt}+k_2((1-b_2)+b_2(\frac{C_s}{C_S}))}.\log\frac{K}{K_t}$	Cluster Model. represents frequency of the document containing term $term_t$ with respect to the cluster C_j and collection ζ
λ	Smoothing Parameter. To control the influence of above two models. 4.3

In order to evaluate $CBT\ BM25$ for both document ranking and clustering algorithms, as we did in the previous chapters for CBT and $CBTV$, we performed two experiments. First, we ran the clustering algorithms (K-means and bisecting K-means) with $CBT\ BM25$ and $Okapi\ BM25$. Second, we ran the document ranking algorithm with $CBT\ BM25$ and $Okapi\ BM25$ to evaluate the retrieved document list. The document clustering and ranking algorithms are given below:

K-means algorithm with $CBT\ BM25$

Initially, the K-means algorithm doesn't have any information about the cluster components, so we start the algorithm by setting df_{tj} and icf_t to 1. The K-means algorithm 9 uses Equation 6.3 and updates the document and cluster models on each iteration.

The runtime complexity of the traditional K-means algorithm is $O(LNK)$ where L is the total number of iterations in the outer loop, N is the total number of documents in a collection and K is the total number of clusters. Algorithm 9 updates the term weights for all the clusters on each iteration to reflect the changes made in the new clusters, so it takes $O(LNK + LK) = O(LNK)$ since $LK < LNK$.

Bisecting K-means algorithm with $CBT\ BM25$

Like the K-means algorithm in the previous section, df_{tj} and icf_t are set to 1 initially.

The runtime complexity of the bisecting K-means algorithm is linear with the number of documents in the collection ζ. Algorithm 10 updates the term weights for all the clusters for K times to reflect the changes made in the new clusters, so it takes $O(N + K^2) = O(N)$ since $K << N$.

Algorithm 9 K-means partitional clustering algorithm with $CBT\ BM25$

Require: An integer $K \geq 1$, Document Collection ζ.

if $K = 1$ **then**
  return ζ
else
  Initialize $l = 0$, $k_1 = 1.2$, $k_2 = 1.2$, $b_1 = 0.75$ and $b_2 = 0.75$
  $\{C_1^{(0)}, \cdots, C_K^{(0)}\} \leftarrow RANDOMCLUSTERS(\zeta, K)$
  repeat
    for all $d_i \in \zeta, i\colon 1 \ldots N$ **do**
      $m = \arg\min_j |c_j - d_i|$
      $C_m^{(l+1)} \leftarrow C_m^{(l+1)} \cup d_i$
    end for
    $l \leftarrow l + 1$
    $w_{itj} = (1-\lambda)\frac{(k_1+1)tf_{it}}{tf_{it}+k_1((1-b_1)+b_1(\frac{D_l}{D_L})}.\log\frac{N}{N_t} + (\lambda)\frac{(k_2+1)df_{jt}}{df_{jt}+k_2((1-b_2)+b_2(\frac{C_s}{C_S})}.\log\frac{K}{K_t}$; for each term $term_t$ in a document d_i of a cluster $C_j^{(t)}$, $t\colon 1 \ldots T$, $i\colon 1 \ldots N$, $j\colon 1 \ldots K$
    for $j = 1$ **to** K **do**
      $c_j \leftarrow \frac{1}{|C_j^{(l)}|}\sum_{d_i \in C_j^{(l)}} d_i$
    end for
  until No change in K centroids
  return $\{C_1^{(l)}, \cdots, C_K^{(l)}\}$
end if

Algorithm 10 Bisecting K-means hierarchical divisive clustering algorithm with $CBT\ BM25$

Require: An integer $K \geq 1$, Document Collection ζ.

if $K = 1$ **then**
  return ζ
else
  Initialize $C \leftarrow \{\zeta\}$
  for $k = 1$ **to** $K - 1$ **do**
    $L \leftarrow PICK - LARGEST - CLUSTER(C)$
    $\{C_1, C_2\} \leftarrow K - MEANS(L, 2)$
    $C \leftarrow C \cup \{C_1, C_2\}$
    $w_{itj} = (1-\lambda)\frac{(k_1+1)tf_{it}}{tf_{it}+k_1((1-b_1)+b_1(\frac{D_l}{D_L})}.\log\frac{N}{N_t} + (\lambda)\frac{(k_2+1)df_{jt}}{df_{jt}+k_2((1-b_2)+b_2(\frac{C_s}{C_S})}.\log\frac{K}{K_t}$; for each term $term_t$ in a document d_i of a cluster $C_j^{(t)}$ in C, $t\colon 1 \ldots T$, $i\colon 1 \ldots N$, $j\colon 1 \ldots K$
  end for
  return C
end if

$CBT\ BM25$ Document Ranking Algorithm

The document ranking algorithm 11 uses $CBT\ BM25$ shown in Equation 6.4.

Algorithm 11 Document Ranking Algorithm using $CBT\ BM25$

Require: Query q, Cluster set $\{C_1, C_2, \cdots, C_K\}$, and Document Collection ζ.

1: $Score[N] = 0.0$
2: **for all** term $term_t$ in query q **do**
3: **for all** document d_i of cluster C_j in collection ζ **do**
4: $w_{itj} \leftarrow (1-\lambda)\frac{(k_1+1)tf_{it}}{tf_{it}+k_1((1-b_1)+b_1(\frac{D_l}{D_L})}.\log\frac{N}{N_t} + (\lambda)\frac{(k_2+1)df_{jt}}{df_{jt}+k_2((1-b_2)+b_2(\frac{C_s}{C_S})}.\log\frac{K}{K_t}$
5: $Score[i] \leftarrow Score[i] + w_{itj}$
6: **end for**
7: **end for**
8: $R \leftarrow SORT(\zeta, Score[])$
9: **return** R

6.2 Data Collections Used For $CBT\ BM25$

We performed two experiments to compare $Okapi\ BM25$ with $CBT\ BM25$, for the document clustering and ranking. For the first experiment with the document clustering algorithms (K-means and bisecting K-means), we used the TREC [TRE99], WebACE [Lan] and Reuters-21578 [Lew99] data collections. TR11, TR12, TR23, TR31, TR45, FBIS, HITECH, LA1 and LA2 collections are taken from TREC-5, TREC-6 and TREC-7. RE0 and RE1 data sets are from Reuters-21578 collection.

Table 6.2: Data sets used for both the K-means and Bisecting K-means algorithms (with $CBT\ BM25$).

Data set	Collection	# of Doc	# of Class
FBIS	TREC	2463	17
HITECH	TREC	2301	6
K1A	WEBACE	2340	20
LA1	TREC	3204	6
LA2	TREC	3075	6
RE0	REUTERS-21578	1504	13
RE1	REUTERS-21578	1657	25
TR11	TREC	414	9
TR12	TREC	313	8
TR23	TREC	204	6
TR31	TREC	927	7
TR45	TREC	690	10
WAP	WEBACE	2000	20

In the second experiment for evaluating the document ranking algorithm with *Okapi* and *CBT BM*25, we used two data sets: Genomic and Classic3. Classic3 data set is made from three data sets: MED, CRAN and CISI. Each data set contains a query set to evaluate the document ranking model.

Table 6.3: Data sets used for *CBT BM*25 Document Ranking Model.

Data set	# of Doc	Query Set
Genomic	42,251	1 - 50
AP89	84,640	1 - 50
Classic3 (MED)	3893	1 - 30
Classic3 (CRAN)	3893	1 - 50
Classic3 (CISI)	3893	1 - 50

For all the data sets shown in Tables 6.2 and 6.3, we removed the stop words and stemmed using the Porter stemming algorithm [Por80].

6.3 Experimental Results

In our first experiment with the document clustering algorithm, we used both the K-means and bisecting K-means clustering algorithms to compare *Okapi* and *CBT BM*25. The clusters generated by these algorithms are evaluated using the entropy, F-Measure, purity, mutual information and normalized mutual information.

Since the K-means algorithm is unstable and highly dependent on the initial seeds chosen, we ran the algorithm for 10 times on each data set for both *Okapi* and *CBT BM*25. We measured the entropy, F-Measure, purity, mutual information and normalized mutual information for each run. The average of these results measured for the 10 runs is used for the comparison: average entropy, average F-Measure, average purity, mutual information and normalized mutual information.

Table 6.4 shows the average entropy, average F-Measure, average purity, mutual information and normalized mutual information computed for the K-means clustering algorithm with *Okapi* and *CBT BM*25 term weighting schemes.

We repeated the same experiment for the bisecting K-means algorithm. From the experiment on the K-means and bisecting K-means document clustering algorithms, we found that *CBT BM*25 performs much better than *Okapi BM*25 term weighting scheme. Most of the results in *CBT BM*25 computed for evaluating the generated clusters are far better than that of *Okapi BM*25.

Table 6.5 shows the average entropy, average F-Measure, average purity, mutual information and normalized mutual information computed for the bisecting K-means clustering algorithm with *Okapi* and *CBT BM*25 term weighting schemes.

Table 6.4: K-means clustering algorithm - Avg. Entropy, Avg. F-Measure, Avg. Purity, MI and NMI measured for *CBT BM*25 and *Okapi BM*25.

Data Source	*CBT BM*25					*Okapi BM*25				
	Avg. Entropy	Avg. F-Measure	Avg. Purity	MI	NMI	Avg. Entropy	Avg. F-Measure	Avg. Purity	MI	NMI
FBIS	**1.1751**	**0.4233**	**0.6253**	1.2383	0.4371	1.1809	0.3690	0.5991	**1.2624**	**0.4456**
HITECH	**1.2356**	**0.4281**	**0.5100**	**0.4365**	**0.2436**	1.2526	0.4161	0.4950	0.4211	0.2350
K1A	**1.1830**	**0.4144**	**0.6214**	**1.3655**	**0.4558**	1.6931	0.3206	0.4758	0.8975	0.2996
LA1	**0.8171**	**0.6456**	**0.7109**	**0.8732**	**0.4873**	1.3035	0.4265	0.4874	0.3873	0.2161
LA2	**0.8114**	**0.6354**	**0.6967**	**0.8682**	**0.4845**	1.2590	0.4333	0.4934	0.4216	0.2353
RE0	**0.9934**	**0.3435**	**0.6501**	**0.8175**	**0.3187**	1.2120	0.2797	0.5708	0.6137	0.2393
RE1	1.3550	0.3513	0.5920	1.3139	0.4082	**1.2041**	**0.4091**	**0.6356**	**1.4550**	**0.4520**
TR11	**0.7808**	**0.4970**	**0.7413**	**1.1074**	**0.5040**	1.7408	0.1534	0.3609	0.2962	0.1348
TR12	**1.2415**	**0.4193**	**0.5435**	**0.6991**	**0.3362**	1.8185	0.1486	0.3201	0.2260	0.1087
TR23	**1.0721**	**0.3430**	**0.5172**	0.2008	0.1334	1.3108	0.2514	0.5	**0.2549**	**0.1423**
TR31	**1.1344**	**0.3012**	**0.5696**	**0.4268**	**0.2194**	1.3908	0.1680	0.4497	0.1869	0.0961
TR45	**1.0358**	**0.5449**	**0.6565**	**1.0209**	**0.4434**	1.7017	0.2444	0.3729	0.4476	0.1944
WAP	**1.2460**	**0.3966**	**0.5977**	**1.3319**	**0.4446**	1.6950	0.3184	0.4766	0.9399	0.3138

Table 6.5: Bisecting K-means clustering algorithm - Avg. Entropy, Avg. F-Measure, Avg. Purity, MI and NMI measured for *CBT BM*25 and *Okapi BM*25 term weighting schemes.

Data Source	*CBT BM*25					*Okapi BM*25				
	Avg. Entropy	Avg. F-Measure	Avg. Purity	MI	NMI	Avg. Entropy	Avg. F-Measure	Avg. Purity	MI	NMI
FBIS	1.2701	0.4402	0.5935	1.1368	0.4012	**1.2114**	**0.4501**	**0.6041**	**1.2035**	**0.4248**
HITECH	**1.1373**	**0.4890**	**0.5552**	**0.5353**	**0.2988**	1.1950	0.4623	0.5200	0.4785	0.2671
K1A	**1.0710**	**0.4803**	**0.6652**	**1.4421**	**0.4814**	1.4766	0.3847	0.5368	1.0763	0.3593
LA1	**0.8188**	**0.6300**	**0.7136**	**0.8731**	**0.4873**	1.3131	0.4121	0.4729	0.3795	0.2118
LA2	**0.8644**	**0.6165**	**0.6852**	**0.8163**	**0.4556**	1.1933	0.4986	0.5329	0.4896	0.2732
RE0	**0.9661**	**0.4297**	**0.6733**	**0.8433**	**0.3288**	1.0928	0.3603	0.6119	0.7240	0.2822
RE1	1.2978	0.3642	0.6187	1.3386	0.4159	**1.2255**	**0.3919**	**0.6361**	**1.4148**	**0.4395**
TR11	**0.8249**	**0.5194**	**0.7236**	**1.0087**	**0.4591**	1.4804	0.2768	0.4655	0.4282	0.19488
TR12	**1.2435**	**0.4638**	**0.5666**	**0.6414**	**0.3085**	1.5932	0.3018	0.3997	0.3476	0.1671
TR23	**1.0250**	**0.4698**	**0.6373**	**0.4386**	**0.2448**	1.2889	0.2353	0.5029	0.2570	0.14343
TR31	**0.8109**	**0.4700**	**0.6939**	**0.7252**	**0.3727**	1.0704	0.3194	0.5905	0.4745	0.2438
TR45	**0.9894**	**0.5422**	**0.6745**	**1.0564**	**0.4588**	1.2026	0.4836	0.5739	0.8633	0.3749
WAP	**1.14748**	**0.4455**	**0.6380**	**1.3737**	**0.4586**	1.5430	0.3719	0.5101	1.0167	0.3394

Finally, we ran the document ranking algorithm with *Okapi* and *CBT BM*25. *Okapi BM*25 function is highly parametrized and needs a lot of tuning. Since we used the same representation for *CBT BM*25, it has more parameters than *Okapi*. In this experiment, we used the same parameter values for both the document and cluster models. But we think that tuning these parameters with *CBT BM*25 will give better results than the one shown in this thesis.

We used MAP, P@10 and P@100 measures to evaluate the ranked list of documents for the query sets. In addition, we have shown the percentage improvement achieved through *CBT BM*25 compared to the document ranking model with *Okapi BM*25.

Table 6.6: MAP evaluation measure for Document Ranking Model.

Data set	# of Clusters ($K = \sqrt{N}$)	MAP		Improvement (%)
		*CBT BM*25	*Okapi BM*25	
Genomic	206	0.3280	0.3045	+7.72
AP89	291	0.1720	0.1619	+6.24
Classic3 (MED)	62	0.5513	0.5383	+2.42
Classic3 (CRAN)	62	0.3511	0.3587	-2.12
Classic3 (CISI)	62	0.1626	0.1548	+5.04

Table 6.7: P@10 evaluation measure for Document Ranking Model.

Data set	# of Clusters ($K = \sqrt{N}$)	P@10		Improvement (%)
		*CBT BM*25	*Okapi BM*25	
Genomic	206	0.4100	0.3320	+23.49
AP89	291	0.2106	0.2170	-2.95
Classic3 (MED)	62	0.6667	0.6367	+4.71
Classic3 (CRAN)	62	0.2600	0.2760	-5.80
Classic3 (CISI)	62	0.3244	0.3200	+1.38

In this experiment, each data set in Table 6.3 is clustered in advance. The generated clusters, *static clusters*, will be used in the cluster model for computing the score/rank for each document in the collection ζ. These static clusters can be generated using any document clustering algorithm. In our experiment, we used the traditional K-means partitional clustering algorithm with $K = \sqrt{N}$ as the number of clusters. We have also observed that the results obtained depends on the quality of the static clusters generated by a clustering algorithm.

Table 6.8: P@100 evaluation measure for Document Ranking Model.

Data set	# of Clusters ($K = \sqrt{N}$)	P@100		Improvement (%)
		CBT BM25	*Okapi BM25*	
Genomic	206	0.3036	0.2704	+12.28
AP89	291	0.1343	0.1251	+7.35
Classic3 (MED)	62	0.1830	0.1830	-
Classic3 (CRAN)	62	0.0536	0.0532	+0.75
Classic3 (CISI)	62	0.1589	0.1511	+5.16

All the evaluation measures have shown that using *CBT BM25* for document ranking has improved the final results. The results can be expected to improve, if we use hierarchical clustering for generating the static clusters.

Chapter 7 Conclusion

In this thesis, we presented a family of cluster-based term weighting schemes focusing on their application in document clustering and ranking. They are CBT, $CBTV$ and $CBT\ BM25$. CBT is the basic form of the cluster-based term weighting scheme, which uses intra- and inter-cluster components for incorporating cluster information in the term weights. In order to experiment this term weighting scheme for document clustering, we used both partitional (K-means) and hierarchical (bisecting K-means) algorithms to show that CBT can improve the clusters generated by any type of document clustering.

The experiments based on CBT have shown better results compared to the traditional and widely used term weighting scheme, $TF - IDF$. We used the average entropy, average F-Measure, average purity, mutual information and normalized mutual information as our evaluation measures for the generated clusters.

Based on the results from CBT, we modified the basic structure of the cluster-based term weighting scheme. We introduced $CBTV$, which includes two models (document and cluster models) whose influence is controlled by a smoothing parameter. We used the Bayesian smoothing with Dirichlet prior for the smoothing parameter in $CBTV$.

The document model tries to find the importance of a term within a document and a collection, whereas the cluster model finds the importance of a term within a cluster and a collection. The cluster model tries to find the importance of an unseen term in a document based on the statistics from the cluster of that document. The structure of this new variant resembles a probabilistic model and has shown experimentally to outperform the $TF - IDF$. This motivated us to use $CBTV$ for document ranking.

We compared the $TF-IDF$ and $CBTV$ for document ranking. The experimental results have shown significant improvements in the retrieved set returned for a given query. We have achieved up to 60% for some data sets. We used MAP, P@10 and P@100 measures to evaluate the ranked list of documents returned for a query.

After our experiment with document ranking using $CBTV$, we tried our CBT approach to one of the best-known probabilistic ranking models, *Okapi* $BM25$. We proposed another ranking function based on $CBTV$ and *Okapi* $BM25$, $CBT\ BM25$. $CBT\ BM25$ has the same structure as $CBTV$ with the document and cluster models controlled by a smoothing parameter. We experimented with both *Okapi* and $CBT\ BM25$ models for document clustering (K-means and bisecting K-means document clustering algorithms) to compare their final results. $CBT\ BM25$ outperforms *Okapi* in most cases. The experimental results from document ranking with *Okapi* and $CBT\ BM25$ have shown that $CBT\ BM25$ gives considerable improvement in the retrieved set.

Document ranking using $CBTV$ and $CBT\ BM25$ needs a set of static clusters to compute the score/rank for each document. In our experiment, we used the K-means partitional clustering for generating these static clusters.

7.1 Future Works

Quality of Static Clusters

From our experiments, we have found that the quality of the static clusters used for $CBTV$ and $CBT\ BM25$ have a significant impact on the retrieved set for a given query. Using the clustering algorithm such as UPGMA can improve the final results in document ranking since UPGMA generates better clusters than the K-means and bisecting K-means clustering algorithms.

Parameter Tuning

The parameter values used in $CBT\ BM25$ depend on the collection used. Since we have used the same parameter values for $CBT\ BM25$ and $Okapi\ BM25$, tuning these parameters can improve the final results from document ranking. We need separate development and test collections, with their query sets, for experimenting this approach.

Smoothing Parameter

In $CBTV$ and $CBT\ BM25$, we used the Bayesian smoothing with Dirichlet prior for computing the smoothing parameter. Comparing the performance of CBT models with different smoothing methods such as Jelinek-Mercer smoothing, Bayesian smoothing with Dirichlet Prior, absolute discounting, Laplace smoothing, Katz smoothing and Good-Turing estimation will be an interesting approach to focus on.

Bibliography

[AK02] Hanan Ayad and Mohamed S. Kamel. Topic Discovery from Text Using Aggregation of Different Clustering Methods. In *Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, pages 161–175, May 2002.

[BMWC98] Chris Buckley, Mandar Mitra, Janet Walz, and Claire Cardie. Using clustering and superconcepts within smart: Trec 6. In *The Sixth Text Retrieval Conference (Trec-6). Nist Special Publication 500-240, National Institute Of Standards And Technology*, pages 107–124, 1998.

[Bol98] Daniel Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, December 1998.

[Buc] Chris Buckley. New retrieval approaches using smart : Trec 4. pages 25–48.

[CK99] Erica Chisholm and Tamara G. Kolda. New term weighting formulas for the vector space method in information retrieval. Technical report, 1999.

[CKPT92] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '92*, pages 318–329, New York, New York, USA, June 1992. ACM Press.

[CMS09] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison Wesley, 1 edition, February 2009.

[CO05] Ronan Cummins and Colm O'Riordan. Evolving general term-weighting schemes for information retrieval: Tests on larger collections. *Artif. Intell. Rev.*, 24:277–299, November 2005.

[CS96] Peter Cheeseman and John Stutz. *Bayesian Classification (AutoClass): Theory and Results*, volume 180, pages 153–180. MIT Press, 1996.

[DHZ+01] Chris H. Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 107–114, Washington, 2001. IEEE Computer Society.

[ELL01] Brian S Everitt, Sabine Landau, and Morven Leese. *Cluster Analysis*, volume 33 of *Social Science Research Council Reviews of Current Research*. Arnold, 2001.

[GRS98] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE. *ACM SIGMOD Record*, 27(2):73–84, June 1998.

[GRS99] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. In *15th International Conference on Data Engineering*, pages 512 – 521, March 1999.

[HB99] S. Hettich and S. D. Bay. Reuters Transcribed Subset, 1999.

[HXZ+98] Tianming Hu, Hui Xiong, Wenjun Zhou, Sam Yuan Sung, and Hangzai Luo. Hypergraph partitioning for document clustering: A summary of results. *Bulletin of the Technical committee on Data Engineering*, 21(1):15 – 22, July 1998.

[JD88] Anil K Jain and Richard C Dubes. *Algorithms for Clustering in Data*. Prentice Hall, 1988.

[Jon81] Karen Sparck Jones. *Information Retrieval Experiment*. Butterworth-Heinemann, Newton, MA, USA, 1981.

[JPD00] Yunjae Jung, Haesun Park, and Ding-Zhu Du. An effective term-weighting scheme for information retrieval. 2000.

[KHK99] G Karypis, E H Han, and V Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[Kin67] B King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 69:86–101, 1967.

[KR90] Leonard Kaufman and Peter J Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Mathematical Statistics. Wiley, 1990.

[LA99] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining KDD 99*, volume 5, pages 16–22. ACM Press, 1999.

[Lan] Ken Lang. 20 Newsgroups Data set.

[LC04] Xiaoyong Liu and W. Bruce Croft. Cluster-based retrieval using language models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 186–193, New York, NY, USA, 2004. ACM.

[LCS97] Dik L. Lee, Huei Chuang, and Kent Seamons. Document ranking and the vector-space model. *IEEE Softw.*, 14:67–75, March 1997.

[Lew99] David D. Lewis. The reuters-21578 text categorization test collection, 1999.

[Mac67] J B MacQueen. Some methods for classification and analysis of multivariate observations. In L M Le Cam and J Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Western Management Science Institute, University of California Press, 1967.

[Mac02] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*, chapter 16, page 496. Cambridge University Press, 2008.

[MZ11] Keerthiram Murugesan and Jun Zhang. Hybrid hierarchical clustering: An experimental analysis. Technical report, University of Kentucky, 2011.

[NH94] R T Ng and J Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the International Conference on Very Large Data Bases*, pages 144–155, 1994.

[Por80] M F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[RJP+06] Joel Reed, Yu Jiao, Thomas Potok, Brian Klump, Mark Elmore, and Ali Hurson. TF-ICF: A New Term Weighting Scheme for Clustering Dynamic Data Streams. In *2006 5th International Conference on Machine Learning and Applications (ICMLA'06)*, pages 258–263. IEEE, December 2006.

[RWBW99] S.E. Robertson, S. Walker, M. Beaulieu, and Peter Willett. Okapi at trec-7: Automatic ad hoc, filtering, vlc and interactive track. *In*, 21:253–264, 1999.

[RZRZ] C S. Robertson, H. Zaragoza, Stephen Robertson, and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond.

[Sal89] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[SB88] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[SBMM96] Amit Singhal, Chris Buckley, Mandar Mitra, and Ar Mitra. Pivoted document length normalization. pages 21–29. ACM Press, 1996.

[SG00] Alexander Strehl and Joydeep Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *HiPC*, pages 525–536, December 2000.

[Sha48] C E Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(4):379–423, 1948.

[SKK00] M Steinbach, G Karypis, and V Kumar. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Department of Computer Science and Engineering University of Minnesota, Citeseer, 2000.

[SM86] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[SS73] P H A Sneath and R R Sokal. *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. A Series of books in biology. Freeman, 1973.

[SW97] Karen Spark Jones and Peter Willett. *Readings in Information Retrieval*. Morgan Kaufmann, 1997.

[TRE99] TREC. Text REtrieval Conference (TREC), 1999.

[TSK05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*, volume chapter 8. May 2005.

[Voo85] Ellen M. Voorhees. The cluster hypothesis revisited. In *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '85, pages 188–196, New York, NY, USA, 1985. ACM.

[XW05] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[ZHD+01] Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. *Bipartite graph partitioning and data clustering*. ACM Press, New York, New York, USA, October 2001.

[ZKF05] Ying Zhao, George Karypis, and Usama Fayyad. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168, March 2005.

[ZL04] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22:179–214, April 2004.

[ZWLX09] Chengzhi Zhang, Huilin Wang, Yao Liu, and Hongjiao Xu. Document Clustering Description Extraction and Its Application. In *Proceedings of the 22nd International Conference on Computer Processing of Oriental Languages. Language Technology for the Knowledge-based Economy*, ICCPOL '09, pages 370–377, Berlin, Heidelberg, 2009. Springer-Verlag.

[ZZH07] Xiaohua Zhou, Xiaodan Zhang, and Xiaohua Hu. Dragon Toolkit: Incorporating Auto-Learned Semantic Knowledge into Large-Scale Text Retrieval and Mining. In *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, pages 197–201. IEEE, October 2007.

VITA

Date of Birth - June 16, 1987

Place of Birth - Tamil Nadu, India

Education Bachelor's Degree in Computer Science and Engineering, Anna University, India (2008).

- **Keerthiram Murugesan**