Term Weighting Evaluation in Bipartite Partitioning for Text Clustering

Chao Qu[1,2], Yong Li[1], Jun Zhu[1,2], Peican Huang[1], Ruifen Yuan[1], and Tianming Hu[1,3]

[1] Dongguan University of Technology, China
[2] Zhongshan University, China
[3] East China Normal University, China
tmhu05@gmail.com

Abstract. To alleviate the problem of high dimensions in text clustering, an alternative to conventional methods is bipartite partitioning, where terms and documents are modeled as vertices on two sides respectively. Term weighting schemes, which assign weights to the edges linking terms and documents, are vital for the final clustering performance. In this paper, we conducted an comprehensive evaluation of six variants of tf/idf factor as term weighting schemes in bipartite partitioning. With various external validation measures, we found $tfidf$ most effective in our experiments. Besides, our experimental results also indicated that df factor generally leads to better performance than tf factor at moderate partitioning size.

1 Introduction

The high dimension in text clustering is a major difficulty for most probabilistic methods such as Naive Bayes [1] and AutoClass [2]. To circumvent this problem, graph-theoretic techniques have been considered for clustering [3]. They model the document similarity by a graph whose vertices correspond to documents and weighted edges give the similarity between vertices. Graphs can also model terms as vertices and similarity between terms is based on documents in which they co-occur. Partitioning the graph yields a clustering of terms, which is assumed to be associated with similar concepts [4]. The duality between document and term clustering can also be naturally modeled using a bipartite, where documents and terms are modeled as vertices on two sides respectively and only edges linking different types of vertices are allowed in the graph [5]. Finding an optimal partitioning in such a bipartite gives a co-clustering of documents and terms, with the expectation that documents and terms in the same cluster are related to the same topic. In addition, bipartite graphs have also been used to model other relationships, such as (documents, concepts)[6] and (authors, publications)[7].

Since the general partitioning goal is to minimize the edge cut, term weighting schemes, which assign weights to edges linking term and document vertices, is a vital step to the final clustering performance. Many researchers have studied text clustering based on different term weighting schemes or different criterion

H. Li et al. (Eds.): AIRS 2008, LNCS 4993, pp. 393–400, 2008.

functions using conventional probabilistic methods [8,9,10,11]. For instance, the authors of [9] pointed out that it is the text representation schemes that dominate the clustering performance rather than the kernel functions of support vector machines (SVM). In other words, choosing an appropriate term weighting scheme is more important than choosing and tuning kernel functions of SVM for text categorization. However, to the best of our knowledge, there is little work on comparing weighting schemes for graph based text clustering, not to mention bipartite partitioning. For this purpose, we concentrate on the comparison of various term weighting schemes in bipartite based text clustering. Specifically, we provide a comprehensive experimental evaluation with real world document datasets from various sources, and with various external validation measures.

Overview. The rest of this paper is organized as follows. Section 2 describes the necessary background. Section 3 provides the results of extensive experiments. Finally, we draw conclusions in Section 4.

2 Background

In this section, we first describe the overview of bipartite based text clustering. Then we introduce the term weighting schemes that will be compared in this paper.

2.1 Bipartite Generation

To apply clustering algorithms, a document data set is usually represented by a matrix. First we extract from documents unique content-bearing words as features, which involves removing stopwords and those with extreme document frequencies. More sophisticated techniques use support or entropy to filter words further. Then each document is represented as a vector in this feature space. With rows for documents and columns for terms, the matrix A's non-zero entry A_{ij} indicates the presence of term w_j in document d_i, while a zero entry indicates an absence.

A graph $G = (V, E)$ is composed of a vertex set $V = \{1, 2, ..., |V|\}$ and an edge set $\{(i, j)\}$ each with edge weight E_{ij}. The graph can be stored in an adjacency matrix M, with entry $M_{ij} = E_{ij}$ if there is an edge (i, j), $M_{ij} = 0$ otherwise. Given the $n \times m$ document-term matrix A, the bipartite graph $G = (V, E)$ is constructed as follows. First we order the vertices such that the first m vertices index the terms while the last n index the documents, so $V = V_W \cup V_D$, where V_W contains m vertices each for a term, and V_D contains n vertices each for a document. Edge set E only contains edges linking different kinds of vertices, so the adjacency matrix M may be written as $\begin{pmatrix} 0, A \\ A^T, 0 \end{pmatrix}$.

2.2 Bipartite Partitioning

Given a weighted graph $G = \{V, E\}$ with adjacency matrix M, clustering the graph into K parts means partitioning V into K disjoint clusters of vertices

$V_1, V_2, ..., V_K$, by cutting the edges linking vertices in different parts. The general goal is to minimize the sum of the weights of those cut edges. Formally, the cut between two vertex groups V_1 and V_2 is defined as $cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} M_{ij}$. Thus the goal can be expressed as $min_{\{V_1, V_2, ..., V_K\}} \sum_{k=1}^{K} cut(V_k, V - V_k)$. To avoid trivial partitions, often the constraint is imposed that each part should be roughly balanced in terms of part weight $wgt(V_k)$, which is often defined as sum of its vertex weight. That is, $wgt(V_k) = \sum_{i \in V_k} wgt(i)$. The objective function to minimize becomes $\sum_{k=1}^{K} \frac{cut(V_k, V - V_k)}{wgt(V_k)}$. Given two different partitionings with the same cut value, the above objective function value is smaller for the more balanced partitioning.

In practice, different optimization criteria have been defined with different vertex weights. The ratio cut criterion [12], used for circuit partitioning, defines $wgt(i) = 1$ for all vertices i and favors equal sized clusters. The normalized cut criterion [13], used for image segmentation, defines $wgt(i) = \sum_j M_{ij}$. It favors clusters with equal sums of vertex degrees, where vertex degree refers to the sum of weights of edges incident on it.

Finding a globally optimal solution to such a graph partitioning problem is in general NP-complete [14], though different approaches have been developed for good solutions in practice [15,16]. Here we employ Graclus [16], a fast kernel based multilevel algorithm, which involves coarsening, initial partitioning and refinement phases. As for the graph partitioning criterion used in Graclus, we tried both the normalized cut criterion and the ratio cut criterion. We found the former always produces better results, possibly for the following reasons. First, our datasets are highly imbalanced, which makes unreasonable the constraint of equal sized clusters by the ratio cut criterion. Second we find that sometimes it yields clusters of pure word vertices, which makes it impossible to determine the number of document clusters (clusters containing the document vertices) beforehand. Those terms with low frequencies are likely to be isolated together, since few edges linking outside are cut. As for the normalized cut criterion that tries to balance sums of vertex degrees in each cluster, the resultant clusters tend to contain both document and term vertices. So in this paper we only report results from the normalized cut criterion.

2.3 Term Weighting

Term weighting schemes determine the value of non-zero entry A_{ij} in the document-term matrix when term w_j appears in document d_i. Two frequencies are commonly used. Term frequency tf_{ij} denotes the raw frequency of term w_j in document d_i. Inverse document frequency $idf_j = log(n/n_j)$ considers the discriminating power of w_j, where n_j is the number of documents that contain w_j. In this paper, we compared six term weighting schemes listed in Table 1. Most of these term weighting schemes have been widely used in information retrieval and text categorization. The first four term weighting schemes are different variants of tf factor. The last two incorporate idf. According to a recent

Table 1. Term weighting schemes

name	description
$binary$	1 for presence and 0 for absence
tf	raw tf
$logtf$	$log(1+tf)$
itf	$1-1/(1+tf)$
idf	$log(n/n_j)$
$tfidf$	$tf \times idf$

study of text classification with SVM [11], although the first four schemes relate with term frequency alone, all of them show competitive performance with other sophisticated schemes except *binary*. The *idf* factor, taking the collection distribution into consideration, does not improve the terms discriminating power with SVM. The $tfidf$ factor, combining both term and document frequencies, usually yields best results in query based document retrieval [17].

3 Experimental Evaluation

In this section, we present an extensive experimental evaluation of various term weighting schemes. First we introduce the experimental datasets and cluster validation criteria, then we report comparative results.

3.1 Experimental Datasets

For evaluation, we selected 10 real data sets from different domains used in [10]. The RE0 and RE1 datasets are from the Reuters-21578 text categorization test collection Distribution 1.0. The datasets K1a, K1b and WAP are from the WebACE project; each document corresponds to a web page listed in the subject hierarchy of Yahoo. In particular, K1a and K1b contain the same data but K1a's class labels are at a finer level. The datasets TR31 and TR41 were derived from the TREC collection. The LA1 and LA2 datasets were obtained from articles of the Los Angeles Times that was used in TREC-5. The FBIS dataset is from the Foreign Broadcast Information Service data of TREC-5. For all data sets, we used a stoplist to remove common words, stemmed the remaining words using Porter's suffix-stripping algorithm, and discard those with very low document frequencies. Some characteristics of them are shown in Table 2.

3.2 Validation Measures

Because the true class labels of documents are known, we can measure the quality of the clustering solutions using external criteria that measure the discrepancy between the structure defined by a clustering and what is defined by the class labels. First we compute the confusion matrix C with entry C_{ij} as the number of documents from true class j that are assigned to cluster i. Then we calculate the following four measures: normalized mutual information(NMI), conditional entropy(CE), error rate(ERR) and F-measure.

Table 2. Characteristics of data sets

data	re0	re1	k1a	k1b	wap	tr31	tr41	la1	la2	fbis
#doc	1504	1657	2340	2340	1560	927	878	3204	3075	2463
#word	2886	3758	4707	4707	8460	10128	7454	6188	6060	2000
#class	13	25	20	6	20	7	10	6	6	17
MinClass	11	13	9	60	5	2	9	273	248	38
MaxClass	608	371	494	1389	341	352	243	943	905	506
min/max	0.018	0.035	0.018	0.043	0.015	0.006	0.037	0.29	0.274	0.075
source	Reuters-21578		WebACE			TREC-6,7		TREC-5		

NMI and CE are entropy based measures. The cluster label can be regarded as a random variable with the probability interpreted as the fraction of data in that cluster. Let T and C denote the random variables corresponding to the true class and the cluster label, respectively. The two entropy-based measures are defined as $NMI = \frac{H(T)+H(C)-H(T,C)}{\sqrt{H(T)H(C)}}$, $CE = H(T|C) = H(T,C) - H(C)$, where $H(X)$ denotes the entropy of X and $\log_2$ is used here in computing entropy. NMI measures the shared information between T and C and it reaches the maximal value of 1 when they are the same. CE tells the information remained in T after knowing C and it reaches the minimal value of 0 when they are identical. Error rate $ERR(T|C)$ computes the fraction of misclassified data when all data in each cluster is classified as the majority class in that cluster. It can be regarded as a simplified version of $H(T|C)$.

F-measure combines the precision and recall concepts from information retrieval [17]. We treat each cluster as if it were the result of a query and each class as if it were the desired set of documents for a query. We then calculate the recall and precision of that cluster for each given class as follows: $R_{ij} = C_{ij}/C_{+j}$, $P_{ij} = C_{ij}/C_{i+}$, where C_{+j}/C_{i+} is the sum of jth column/i-th row, i.e., j-th class size /i-th cluster size. Note that C_{+j} could be larger than the true size of class j if some documents from it appear in more than one cluster. F-measure of cluster i and class j is then given by $F_{ij} = \frac{2R_{ij}P_{ij}}{P_{ij}+R_{ij}}$. The overall value for the F-measure is a weighted average for each class, $F = \frac{1}{n}\sum_j C_{+j} max_i\{F_{ij}\}$, where n is the total sum of all elements of matrix C. F-measure reaches its maximal value of 1 when the clustering is the same as the true classification.

3.3 Clustering Results

Let c in cK denote the number of partitions we set. The six term weighting schemes are evaluated in terms of the four validation measures at $5K, 10K, 15K$ and $20K$. The detailed results are shown in Table 3. NMI and F are preferred large while ERR and CE are preferred small. For each setting, the best results are highlighted in bold numbers. One can see in most cases $tfidf$ gives the best results and *binary* performs worst. Although tf, $logtf$ and itf perform best in certain settings, their gap between $tfidf$ is not significant. The last six rows of Table 3 give the number of wins over all measures at four levels of clustering granularity, respectively. The superiority of $tfidf$ is obvious at three levels.

Table 3. Comparison of clustering results

		5K				10K				15K				20K			
data	wgt	NMI	CE	ERR	F	NMI	CE	ERR	F	NMI	CE	ERR	F	NMI	CE	ERR	F
re0	binary	0.28	1.43	0.48	0.40	0.32	2.15	0.46	0.36	0.32	2.71	0.40	0.34	0.33	2.87	0.39	0.31
	tf	**0.33**	1.40	0.45	0.47	0.31	2.18	0.45	0.36	0.33	2.69	0.39	0.37	0.32	2.95	0.42	0.31
	logtf	0.30	1.28	0.47	0.43	0.30	2.24	0.43	0.35	0.33	2.64	0.40	0.33	0.35	2.82	0.38	0.33
	itf	0.29	1.40	0.51	0.42	**0.35**	**2.07**	**0.39**	**0.40**	**0.37**	**2.43**	**0.37**	**0.39**	**0.39**	**2.77**	**0.35**	**0.36**
	idf	0.29	**1.26**	**0.44**	**0.49**	0.26	2.10	0.48	0.36	0.24	2.72	0.44	0.30	0.27	3.01	0.43	0.29
	tfidf	0.25	1.44	0.47	0.45	0.29	2.15	0.44	0.35	0.31	2.68	0.41	0.33	0.32	2.87	0.43	0.33
re1	binary	0.24	**1.19**	0.66	0.29	0.29	1.67	**0.58**	0.33	0.31	2.36	0.56	0.33	0.33	2.51	0.55	0.34
	tf	0.26	1.46	0.63	0.31	**0.31**	1.68	0.58	**0.36**	0.33	2.37	0.56	0.31	**0.35**	2.40	**0.51**	**0.35**
	logtf	**0.28**	1.36	**0.62**	0.33	0.26	1.93	0.64	0.28	0.31	2.17	0.56	0.32	0.35	2.42	0.53	0.35
	itf	0.25	1.22	0.66	0.30	0.26	1.96	0.60	0.31	0.28	2.28	0.60	0.32	0.33	2.67	0.56	0.33
	idf	0.20	1.27	0.70	0.24	0.26	1.78	0.61	0.31	0.30	2.12	0.57	**0.34**	0.32	2.35	0.55	0.32
	tfidf	0.28	1.34	0.62	**0.36**	0.30	**1.60**	0.60	0.32	**0.33**	**2.12**	**0.56**	0.32	0.35	**2.16**	0.56	0.31
k1a	binary	0.08	0.82	0.75	0.19	0.29	0.93	0.66	0.31	0.24	1.51	0.67	0.32	0.30	1.62	0.64	0.38
	tf	0.11	0.45	0.75	0.21	0.38	0.66	0.63	0.36	0.32	1.18	0.60	0.38	0.39	**1.26**	0.56	0.43
	logtf	0.16	0.39	0.73	0.22	**0.39**	**0.62**	0.63	0.36	0.38	**1.03**	0.58	0.41	0.37	1.32	0.59	0.43
	itf	0.16	**0.37**	0.73	0.22	0.36	0.69	0.62	0.35	0.34	1.10	0.60	0.38	0.37	1.29	0.59	0.42
	idf	**0.31**	0.66	0.65	0.30	0.31	1.19	0.63	0.36	**0.45**	1.32	**0.49**	**0.48**	0.39	1.68	**0.53**	0.44
	tfidf	0.30	0.95	**0.64**	**0.33**	0.35	1.44	**0.59**	**0.39**	0.40	1.30	0.59	0.42	**0.41**	1.56	0.54	**0.44**
k1b	binary	0.07	0.88	0.41	0.49	0.33	1.09	0.30	0.63	0.29	1.64	0.27	0.51	0.34	1.83	0.24	0.54
	tf	0.03	0.59	0.40	0.50	0.39	0.93	0.28	**0.69**	0.29	1.53	0.30	0.60	0.42	1.57	0.22	**0.60**
	logtf	0.06	0.57	0.40	0.49	0.40	**0.89**	0.26	0.68	0.38	**1.38**	0.29	**0.64**	0.42	1.56	0.22	0.58
	itf	0.08	**0.53**	0.40	0.49	0.36	0.96	0.27	0.64	0.33	1.44	0.24	0.57	**0.43**	**1.53**	0.22	0.59
	idf	0.33	0.86	0.33	0.63	0.37	1.34	0.25	0.57	**0.47**	1.71	**0.19**	0.56	0.42	2.04	0.22	0.51
	tfidf	**0.33**	1.14	**0.23**	**0.65**	**0.41**	1.67	**0.20**	0.66	0.45	1.58	0.22	0.60	0.42	1.95	**0.21**	0.50
wap	binary	0.46	0.32	0.61	0.39	0.40	1.40	0.53	0.41	0.39	1.52	0.56	0.38	0.43	1.56	0.49	0.42
	tf	0.46	0.76	0.58	0.45	0.40	1.32	0.56	0.39	0.40	1.37	0.54	0.36	0.43	**1.39**	0.51	0.34
	logtf	0.45	0.82	0.56	**0.46**	0.45	1.17	0.52	0.41	0.41	1.45	0.53	0.38	0.38	1.61	0.55	0.36
	itf	0.48	0.40	0.57	0.43	**0.50**	0.97	0.52	**0.48**	0.43	1.53	0.54	0.42	0.44	1.80	0.46	0.42
	idf	0.47	0.75	**0.55**	0.46	0.50	1.03	**0.50**	0.44	0.50	**1.31**	0.46	**0.48**	0.50	1.62	0.46	**0.48**
	tfidf	**0.48**	**0.23**	0.60	0.42	0.50	**0.73**	0.52	0.44	**0.52**	1.48	**0.44**	0.46	**0.52**	1.65	**0.40**	0.48
tr31	binary	0.23	**0.51**	0.52	0.51	0.21	1.61	0.48	0.49	0.31	1.64	0.39	0.58	0.29	1.90	0.38	0.55
	tf	0.25	0.78	0.51	0.48	**0.52**	**0.68**	**0.28**	**0.72**	**0.49**	**1.02**	**0.28**	**0.67**	0.54	1.27	0.20	0.68
	logtf	0.22	1.00	0.51	0.48	0.44	0.87	0.32	0.69	0.44	1.19	0.29	0.61	0.50	**1.26**	0.22	**0.70**
	itf	0.18	1.00	0.53	0.45	0.41	0.82	0.32	0.66	0.44	1.16	0.29	0.64	0.47	1.34	0.24	0.66
	idf	0.32	0.69	**0.40**	0.54	0.48	1.07	0.29	0.64	0.39	1.75	0.33	0.58	0.51	1.76	0.20	0.60
	tfidf	**0.34**	1.11	0.50	**0.55**	0.51	1.11	0.28	0.64	0.48	1.41	0.28	0.60	**0.56**	1.42	**0.17**	0.69
tr41	binary	0.54	0.89	0.33	0.66	0.57	1.44	0.30	0.63	0.48	1.98	0.32	0.48	0.45	2.25	0.36	0.42
	tf	0.67	0.51	0.28	0.71	0.60	1.36	0.29	0.68	0.61	1.77	**0.20**	0.58	0.59	2.06	0.19	0.55
	logtf	**0.67**	**0.49**	**0.27**	**0.72**	**0.68**	**1.02**	**0.18**	**0.74**	**0.62**	1.67	0.21	**0.58**	0.58	2.04	0.21	0.52
	itf	0.65	0.57	0.29	0.70	0.58	1.32	0.24	0.61	0.57	1.76	0.24	0.57	0.55	2.12	0.24	0.54
	idf	0.64	0.60	0.31	0.67	0.59	1.20	0.30	0.66	0.54	**1.66**	0.28	0.53	0.54	2.00	0.30	0.51
	tfidf	0.58	0.76	0.33	0.64	0.61	1.14	0.26	0.65	0.58	1.67	0.24	0.57	**0.61**	**1.90**	**0.18**	**0.57**
la1	binary	0.01	1.02	0.70	0.32	0.15	1.43	0.60	0.41	0.10	2.00	0.62	0.37	0.15	2.23	0.54	0.42
	tf	0.05	**0.64**	0.68	0.34	0.23	**1.09**	0.58	0.45	0.33	**1.33**	0.42	0.57	0.33	**1.62**	0.42	0.56
	logtf	0.02	0.76	0.70	0.32	0.20	1.17	0.59	0.43	0.27	1.49	0.45	0.54	0.33	1.66	0.43	0.53
	itf	0.02	0.76	0.70	0.32	0.18	1.22	0.61	0.42	0.26	1.51	0.43	0.55	0.32	1.65	0.43	0.53
	idf	0.24	1.14	0.54	0.45	**0.37**	1.22	**0.43**	**0.58**	0.36	1.59	0.40	0.58	0.32	1.99	0.39	0.57
	tfidf	**0.29**	0.91	**0.51**	**0.49**	0.27	1.33	0.53	0.48	**0.39**	1.45	**0.37**	**0.64**	**0.35**	1.82	**0.38**	**0.61**
la2	binary	0.01	0.82	0.70	0.32	0.16	1.28	0.59	0.42	0.10	1.86	0.63	0.38	0.12	2.17	0.58	0.41
	tf	0.03	0.52	0.69	0.33	0.23	1.00	0.57	0.45	0.30	**1.31**	0.44	0.55	0.34	**1.48**	0.41	0.59
	logtf	0.03	0.55	0.69	0.33	0.22	1.04	0.58	0.44	0.17	1.61	0.56	0.42	0.30	1.61	0.44	0.52
	itf	0.02	0.57	0.70	0.33	0.19	1.11	0.59	0.43	0.14	1.68	0.59	0.40	0.21	1.83	0.50	0.49
	idf	0.06	0.94	0.67	0.34	0.39	1.10	0.41	0.60	**0.32**	1.55	**0.41**	**0.57**	0.30	1.92	0.40	0.56
	tfidf	**0.36**	**0.50**	**0.48**	**0.50**	**0.42**	**1.00**	**0.36**	**0.64**	0.32	1.52	0.45	0.52	**0.36**	1.73	**0.40**	**0.60**
fbis	binary	0.38	0.97	0.58	0.44	0.51	1.37	0.41	0.54	0.33	2.59	0.47	0.45	0.32	2.94	0.46	0.40
	tf	0.41	0.81	0.57	0.44	0.51	1.38	0.42	**0.58**	0.50	1.90	0.39	0.52	0.50	2.14	0.37	**0.55**
	logtf	0.32	0.84	0.63	0.39	0.48	1.49	0.44	0.52	**0.51**	**1.68**	0.39	**0.55**	0.50	2.15	**0.37**	0.49
	itf	0.37	0.86	0.58	0.43	0.47	1.51	0.43	0.52	0.48	1.71	**0.38**	0.53	0.49	2.14	0.38	0.52
	idf	**0.45**	**0.77**	**0.49**	**0.48**	0.45	1.36	0.46	0.55	0.46	1.79	0.42	0.51	0.46	2.25	0.40	0.50
	tfidf	0.41	0.86	0.56	0.44	**0.53**	**1.29**	**0.40**	0.55	0.50	1.81	0.41	0.55	**0.51**	**1.91**	0.38	0.53
#wins	binary	2				1				0				0			
	tf	2				9				6				9			
	logtf	7				7				8				3			
	itf	2				6				7				6			
	idf	10				4				**11**				2			
	tfidf	**17**				**13**				8				**20**			

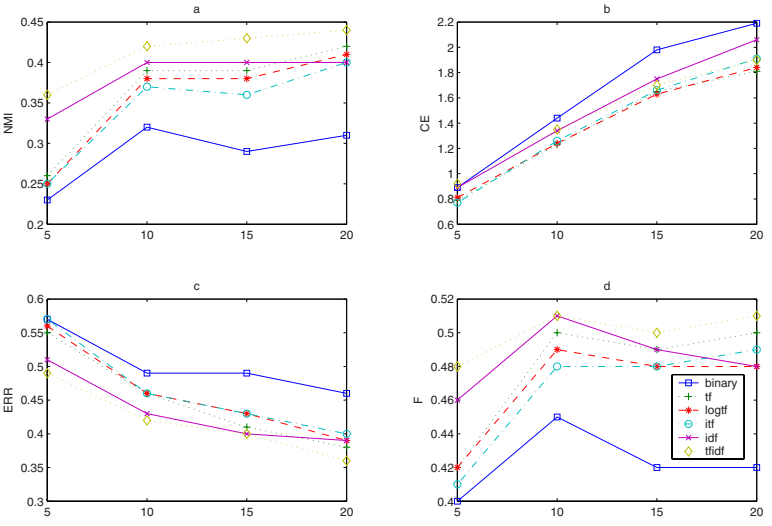

Fig. 1. The average results over the 10 datasets

To give a summary of performance, Figure 1 illustrates the average results over all datasets for each measure. Apparently, $tfidf$ gives the best results on all the measures except CE. Recall that CE is an un-normalized measure, which makes averaging questionable. This indicates that the incorporation of the discriminating factor, idf, really makes a difference. It is confirmed again if we compare the performance by idf and tf alone. One can see at $5K, 10K, 15K$, idf beats tf. It shows that at relatively moderate clustering granularity, the term relevance for the whole document set is more important than that for each individual document. However, as the number of clusters gets larger (e.g., $20K$) and each cluster gets smaller, the term frequency within each document matters.

4 Conclusions

Since the goal of bipartite partitioning is to minimize the edge cut, term weighting schemes are essential for the final clustering performance. In this paper, we provided an extensive comparison of six commonly used schemes. Our experimental results show that $tfidf$ generally yields better performance than other term weighting schemes in terms of various external validation measures. Besides, at moderate clustering granularity, idf is more important than tf. Because the graph partitioning is always subject to the balance constraint, the vertex weighting also plays an important role. For the future work, we plan to investigate the impact of vertex weighting schemes. To capture the full semantics that cannot be represented with single words, another direction is to augment the raw vocabulary with word-sets based on frequent itemsets [18,19] or hypercliques [20,21].

Acknowledgments. This work was partially supported by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry of China, and the Dongguan Foundation of Scientific and Technological Plans.

References

1. Rennie, J.D., Shih, L., Teevan, J., Karger, D.R.: Tackling the poor assumptions of naive bayes text classifiers. In: ICML, pp. 616–623 (2003)
2. Cheeseman, P., Stutz, J.: Bayesian classification (AutoClass): Theory and results. In: KDD, pp. 153–180 (1996)
3. Strehl, A., Ghosh, J., Mooney, R.: Impact of similarity measures on web-page clustering. In: Proc. AAAI: Workshop of Artificial Intelligence for Web Search, pp. 58–64 (2000)
4. Baker, L.D., McCallum, A.: Distributional clustering of words for text classification. In: SIGIR, pp. 96–103 (1998)
5. Dhillon, I.S.: Co-clustering documents and words using bipartite spectral graph partitioning. In: SIGKDD, pp. 269–274 (2001)
6. Yoo, I., Hu, X., Song, I.Y.: Integration of semantic-based bipartite graph representation and mutual refinement strategy for biomedical literature clustering. In: SIGKDD, pp. 791–796 (2006)
7. Sun, J., Qu, H., Chakrabarti, D., Faloutsos, C.: Relevance search and anomaly detection in bipartite graphs. ACM SIGKDD Explorations 7(2), 48–55 (2005)
8. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management 24(5), 513–523 (1988)
9. Leopold, E., Kindermann, J.: Text categorization with support vector machines: How to represent texts in input space? Machine Learning 46(1-3), 423–444 (2002)
10. Zhao, Y., Karypis, G.: Empirical and theoretical comparisons of selected criterion functions for document clustering. Machine Learning 55(3), 311–331 (2004)
11. Lan, M., Tan, C.L., Low, H.B.: Proposing a new term weighting scheme for text categorization. In: AAAI (2006)
12. Hagen, L., Kahng, A.: New spectral methods for ratio cut partitioning and clustering. IEEE Trans. CAD 11, 1074–1085 (1992)
13. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. PAMI 22, 888–905 (2000)
14. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Company, New York (1979)
15. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Scientific Computing 20(1), 359–392 (1998)
16. Dhillon, I.S., Guan, Y., Kulis, B.: A fast kernel-based multilevel algorithm for graph clustering. In: SIGKDD, pp. 629–634 (2005)
17. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Reading (1999)
18. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: SIGMOD, pp. 207–216 (1993)
19. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB, pp. 487–499 (1994)
20. Xiong, H., Tan, P.N., Kumar, V.: Mining strong affinity association patterns in data sets with skewed support distribution. In: ICDM, pp. 387–394 (2003)
21. Xiong, H., Tan, P.N., Kumar, V.: Hyperclique pattern discovery. Data Mining and Knowledge Discovery Journal 13(2), 219–242 (2006)