Term Ranking for Clustering Web Search Results

Fatih Gelgi
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ, 85287, USA
fagelgi@asu.edu

Hasan Davulcu
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ, 85287, USA
hdavulcu@asu.edu

Srinivas Vadrevu
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ, 85287, USA
svadrevu@asu.edu

ABSTRACT

Clustering web search engine results for ambiguous keyword searches poses unique challenges. First, we show that one cannot readily import the frequency based feature ranking to cluster the web search results as in the text document clustering. Next, we present TermRank, a variation of the PageRank algorithm based on a relational graph representation of the content of web document collections. TermRank achieves desirable ranking of discriminative terms higher than the ambiguous terms, and ranking ambiguous terms higher than common terms. We experiment with two clustering algorithms to demonstrate the efficacy of TermRank. TermRank is shown to perform substantially better than frequency based classical methods.

Keywords: Web page clustering, term ranking, PageRank, random walk.

1. INTRODUCTION

During Web search, when keyword queries are *ambiguous* such as ‘apple’, which appears in various contexts such as *Computers* or *Fruit*, it becomes arduous for a user to identify the distinct senses of their search terms and find contextual search results. This problem is further exacerbated when the user is unable to guess additional discriminative keywords, such as ‘ipod’, to filter the matches. For example, there are 544 million results returned by Google search engine[1] for the keyword query ‘apple’. A better way to browse the search results retrieved by a search engine would be to organize them into clusters with their descriptions (like Vivisimo[2], EigenCluster [3], SnakeT [4] and Grouper [15]) to guide the user during her search.

The features in Web document clustering are usually generated from HTML syntax and text in a given collection. Syntactic features are out of the scope of this work; this paper focuses on exploiting most useful terms in a given Web document collection retrieved from an ambiguous keyword search such as ‘apple’.

Clustering has been previously applied on text documents [8]. The common practice is to use all the terms as features [13] in the document collection by generating the term vectors for each document. Especially in large collections using all terms explodes the feature space and gives rise to a well-known problem, “the curse of dimensionality”. It slows down and reduces the quality of clustering dramatically. To avoid this problem, one usual method is to use the top-k best ranked terms as features where k is a reasonable dimension for clustering technique to be used. In that case, the quality of top-k terms needs to be assured by a robust term ranking method.

Broadly used frequency based term ranking methods are TF (term frequency) and TF/IDF (term frequency/inverse document frequency) [13] that penalizes the weights for common keywords that appear in large number of documents such as ‘that’, ‘about’, ‘the’ etc. These measures work well on clustering text documents since text document collections usually contain focused and mostly relevant vocabulary for their terms. However, clustering web search engine results for ambiguous keyword searches poses unique challenges. First, we show that we cannot readily import the frequency based features to cluster the web search results. This is due to the nature of the web search results which (i) usually contain various kinds of irrelevant terms in their navigational aids, advertisements and links to other pages and also due to the observation that (ii) Web pages are usually not as comprehensive as text documents and mostly cover only fragments of the relevant context. Hence, we propose to organize terms found in web search results into three categories: *discriminative terms* that belong to a specific context and strongly related with a distinct sense of the keyword search term, *ambiguous terms* that have many senses, and *common terms* that appear in many distinct contexts of a keyword search term.

Adjusting term weights based on their category is critical for building pure clusters of web search results. For example, for the keyword search term ‘apple’, the term ‘ipod’ is *discriminative* in determining that the web documents matching ‘ipod’ belongs to the ‘computer’ sense of the word. Where as, a *common* term such as ‘contact’ or ‘information’ should not carry much weight during clustering.

In this paper we present TermRank, a variation of the PageRank [11] algorithm based on a relational graph representation of the content of web document collections. TermRank achieves desirable ranking of discriminative terms higher

[1] http://www.google.com

[2] http://www.vivisimo.com

than the ambiguous terms, and ranking ambiguous terms higher than common terms. As indicated in [16], traditional clustering algorithms refine terms after clustering instead of filtering carefully before clustering [9, 14, 5, 7]. However, a good term ranking method improves the quality of the features and helps Web page clustering significantly. We use two clustering algorithms to demonstrate the efficacy of TermRank: K-means, a popular and efficient clustering algorithm, and SCuBA [1], a state-of-art subspace clustering algorithm. We show that performances of classical term ranking methods, TF and TF/IDF, for both K-means and SCuBA are very close. On the other hand, TermRank is shown to perform substantially better than the classical methods for both K-means (purity $\geq +9\%$ and F-measure $\geq +11\%$) and SCuBA (purity $\geq +4\%$ and F-measure $\geq +5\%$).

2. FREQUENCY BASED TERM RANKING

Term ranking is an essential issue in clustering documents. Ranking distinguishing terms higher yields better estimation of similarity between documents and hence higher quality clustering. Standard frequency based term ranking methods in Information Retrieval (IR) are:

- *Term frequency* (TF) is the frequency of a term among all the terms in the Web page collection, and calculated as $TF(t) = \frac{n_t}{n}$ where n_t is the number of occurrences of t in the collection and n is the total number of terms in the collection.

- *Term frequency / inverse document frequency* TF/IDF [13] is a method to reduce the bias of term frequency by penalizing with the document frequency. It is calculated as $TF/IDF(t) = TF(t). \lg \frac{|\mathcal{W}|}{|D(t)|}$ where $D(t)$ is the set of Web pages t appears.

2.1 Analysis on the Web Data

TF/IDF is known to be an effective method for ranking terms in text document collections. However, Web pages are not composed of only raw text and the characteristics of the Web data is different from text documents. We can distinguish between Web and text document collections by making the following key observations:

Observation 1. *Web pages are usually not as comprehensive as text documents and mostly cover only fragments of the relevant context.*

Observation 2. *Almost all terms appearing in a text document can be considered as relevant within the document's context. However, one can observe many context irrelevant terms which might be part of advertisements, headers or footers, and navigation aids such as 'back', 'contact' and 'search' within Web pages.*

Substantial quality difference between text document clustering and Web page clustering is apparent in experimental results of [2, 9]. Such results strongly supports our observations.

We will detail our analysis by investigating the frequency distributions of frequent terms on the Web data. We also observe that, all frequent terms can be grouped into three categories:

- *Discriminative terms*: These terms typically belong to a specific context, and they are strongly related with the context. 'Mac', 'ipod' and 'recipe' are such examples from *apple*[3] data.

- *Ambiguous terms*: These terms appear in more than one context and their degree of relatedness might vary depending on the context. For instance, 'software' and 'computer' appear in both *Computers* and *Video games* categories of the 'apple' data. However, their degree of relatedness would not be weak due to the overlap in the context of categories.

- *Common terms*: These terms appear in many contexts in the data. Unlike the ambiguous terms, they have weak connections with the context that they appear. Some examples of common terms are 'email', 'contact', and 'search'.

In the rest of the paper *important* terms refer to both *discriminative* and *ambiguous* terms.

In Web page clustering, ranking discriminative terms higher than the ambiguous terms, and ranking ambiguous terms higher than the common terms contribute to high accuracy. Since contextual relatedness does not depend on the term frequencies alone and in the Web data common and ambiguous terms have higher frequency characteristics, TF method would mix the ranks of different types of terms. TF/IDF is a good measure for penalizing common and ambiguous terms using inverse document frequency in text collections. However, it becomes ineffective for ranking ambiguous and common terms lower than discriminating terms since they would not appear in sufficiently large numbers due to Observation 1 of Web collections. As a consequence, TF/IDF gets into the same pitfalls as the other frequency based methods such as TF.

To illustrate the similarities of term rankings based on TF and TF/IDF we ranked the top-200 (out of 24,455) terms in the *apple* data. We observed excessive overlap (97%!) between their rankings and similar weighting of the terms.

Consequently, one cannot merely rely on term frequency based measures to rank among the three groups of terms in the preferred orderings described above. Next, we will present a new ranking method in order to obtain the desired term rankings for clustering Web search results.

3. EXTRACTING RELATIONAL GRAPH

A **relational graph** is a weighted undirected graph that captures the co-occurrence relationship information between terms in a given Web page collection. The nodes are the terms in the collection, and the weights on the edges represent the *association strength* between the terms. Formally, we define and initialize a relational graph as follows:

Definition 1 (Relational Graph). *A* ***relational graph*** *$\mathcal{G}$ is a weighted undirected graph where the nodes are the terms and the edge weights are the counts of the corresponding co-occurrence relations in the collection. Assuming w_{ij} as the weight between the terms i and j, w_{ij} is the number of times the edge (i, j) appears in the entire data. The weight of the node i is initialized as the occurrence of the corresponding term in the collection.*

[3] The experimental Web page collection is gathered from `http://dmoz.org` with 'apple' keyword search.

Note that the edges are undirected since association strength between terms is a bidirectional measure. Extracting contextually related terms precisely from Web pages require deep syntactic and semantic analysis. This problem is outside the scope of this paper.

Thus we use a simple heuristic that retrieves only the blocks in which the search keyword appear within each Web page. Here, *block* refers to the text fragments delimited by a set of pre-determined tags such as '<div>', '<span>', '<table>', '<p>', '<ul>' and '<ol>'. Then edges are initialized as the co-occurrence of terms within such blocks. We observed that using the entire page for relation extraction introduces too many irrelevant term associations. For example, terms such as 'copyright', 'search', 'back' are almost always irrelevant to the keyword search context. Our block based heuristic would reduce their association strengths since they would rarely co-occur in the same block with the important terms.

Suppose a relational graph of a Web search result collection is given. We next propose an efficient algorithm to utilize the relational graph to rank the important terms. We also observe that different types of terms presented above have the following association characteristics:

- A discriminative term does not have many neighbors, but important ones, and its associations are strong.
- An ambiguous term has many neighbors and it has strong associations as well as weak ones.
- A common term has many neighbors and mostly its associations are weak.

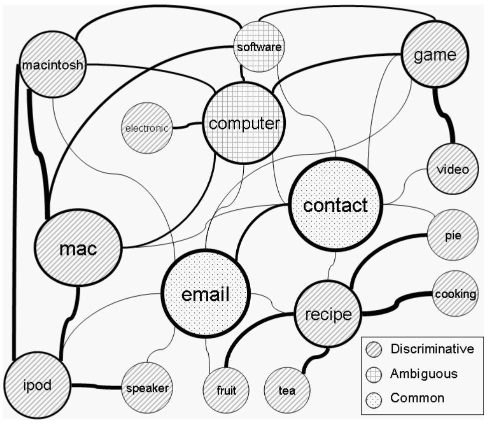

Figure 1: The fragment of the relational graph of *apple* data. Sizes of nodes and thickness of edges are proportional to their term frequencies and association strengths respectively.

A fragment of a relational graph of *apple* data is presented in Figure 1 for illustrative purposes. Discriminative terms such as 'mac' and 'recipe' have neighbors with strong associations. Common terms such as 'contact' and 'email' have many neighbors with weak connections. 'Computer' and 'software' are examples of ambiguous terms that have many neighbors with strong associations as well as weak ones.

Let $\mathcal{W} = \{W_1, W_2, \ldots, W_{|\mathcal{W}|}\}$ be the set of Web pages in the collection. Considering a Web page is the vector of its terms and all pairwise relations between them, extraction of the relational graph of the collection requires $O\left(\sum_{W_i \in \mathcal{W}} |W_i|^2\right)$ time.

4. TERMRANK ALGORITHM

TermRank is a variation of PageRank algorithm that calculates the ranks of the terms in the relational graph. This section gives a brief background on PageRank and its intuitive explanation. Then, we present the TermRank algorithm alongside its justification for term ranking.

4.1 PageRank

PageRank [11] is a method that calculates the importance of the nodes in a link (citation) graph of Web pages. The idea is based on the probability of a "random surfer" visitation to a Web page following the links of Web pages. As indicated in [11], importance of a Web page is considered to be **proportional to the number important sources pointing to that page**. PageRank computes this recursive definition of importance by utilizing a random walk approach over the Web link graph. Random walk propagates the probability of each page to the pages it links to. Given the link graph $\mathcal{G}$ PageRank score for a node is calculated as:

$$PR(i) = \alpha \sum_{j \in \mathcal{N}(i)^-} \frac{PR(j)}{|\mathcal{N}(j)^+|} + (1-\alpha)\frac{1}{|\mathcal{V}(\mathcal{G})|} \qquad (1)$$

where $\mathcal{V}(\mathcal{G})$ is the set of nodes in $\mathcal{G}$, $\mathcal{N}(\cdot)^+$ and $\mathcal{N}(\cdot)^-$ gives the set of neighbors that are connected to the node with their outgoing and incoming links respectively. $(1-\alpha)$ is a decay factor to avoid of rank sinks. Rank sinks [11] are defined to be a set of nodes which have links between themselves but no links to the other nodes. Hence these nodes generate a loop and accumulate rank but they never distribute any rank outside since there is no outgoing edges. That decay factor acts as an exit node with certain probability. An intuitive explanation for the decay factor is given as the jumping probability of a random surfer to other pages.

4.2 TermRank

Originally PageRank operates on a directed graph, and edges have no weights. TermRank adopts the PageRank algorithm to incorporate undirected edges and edge weights. Hence, all edges are considered to be both incoming and outgoing. Additionally, "jumping to a random page" requirement in PageRank is not necessary for TermRank since there are no rank sinks in undirected graphs. Thus decay factor is not included in our formula. Given a relational graph $\mathcal{G}$, TermRank can be calculated as follows:

$$TR(i) = \sum_{j \in \mathcal{N}(i)} \frac{TR(j).w_{ij}}{\sum_{k \in \mathcal{N}(j)} w_{jk}} \qquad (2)$$

where $\mathcal{N}(\cdot)$ represents the set of neighbors of the node. Similar to [11, 6], to compute TermRank, we use a very efficient approximation method which iterates Equation 2:

$$\begin{aligned} TR^{(0)}(i) = & \frac{w_i}{\sum_{j \in \mathcal{V}(\mathcal{G})} w_j} = TF(i) \\ TR^{(t+1)}(i) = & \sum_{j \in \mathcal{N}(i)} \frac{TR^{(t)}(j).w_{ij}}{\sum_{k \in \mathcal{N}(j)} w_{jk}} \end{aligned} . \qquad (3)$$

	TermRank		TF/IDF
	iteration: 0	iteration: 20	
mac	0.1389	0.2600	0.4606
macintosh	0.0663	0.2262	0.2569
game	0.0764	0.1452	0.3666
ipod	0.0928	0.1270	0.3751
video	0.0568	0.1128	0.2549
computer	0.2147	0.1059	0.4679
contact	0.3537	0.0226	0.3864

Table 1: Ranks of terms based on TermRank and TF/IDF. The higher value is the higher rank.

In Equation 3, the ranks of the nodes are initialized with their term frequencies since important terms are assumed to be potentially frequent in the collection. This initialization biases the formula towards the nodes with high term frequencies. Please recall that this set of frequent terms contains important terms as well as common ones.

ItemRank runs until the difference between to iterations is less than δ which is a reasonably small value. TermRank satisfies the extender graph property and hence its convergence is guaranteed [10].

In random walks, a high ranking node needs to satisfy two essential factors: *important neighbors*, *strong connections*. Please note that the number of links of a node is an important factor in PageRank due to its non-weighted (or uniform) links. However, in weighted graphs the strength of the edges are the determining factor. For instance, a node with one edge which has a weight of 10 will receive more rank than a node with 10 edges each of which has a weight of 1 when the rest of their graphs are identical.

The term ranks from higher to lower will be sorted as follows: (i) the nodes with many important neighbors and strong connections, (ii) the ones with some important neighbors and some strong connections, and finally (iii) the ones with many neighbors and many weak connections. Please recall that the term rank orderings presented above would correspond exactly to the desired rankings of discriminative, ambiguous, and common terms.

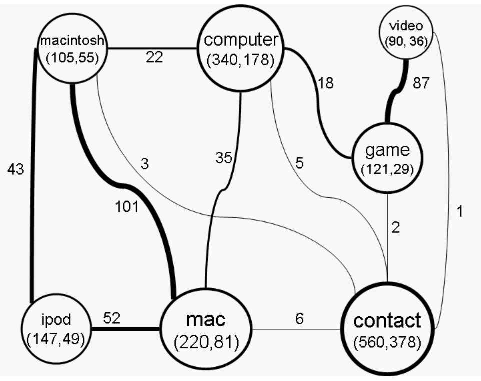

Figure 2: The fragment of the relational graph of *apple* data. The numbers under the node labels represent the term and document counts of the corresponding nodes in the Web page collection. Similarly, the numbers on the edges represent the edge counts.

To illustrate TermRank on an example, Figure 2 which is part of the relational graph of *apple* data is adopted from Figure 1. 'mac', 'macintosh', 'game', 'ipod' and 'video' are discriminative terms, 'computer' is an ambiguous term and 'contact' is a common term. Node weights are initialized as term counts and δ is set to 0.0001. As presented in Table 1, initial ranks of the terms are TF values which gives a mixed ranking in terms of discriminative, ambiguous and common terms. TermRank converges in 20 iterations and it results with the desirable ranking of the terms; first five are the discriminative terms, then the next two are the ambiguous terms and finally the common term. Conversely, 'computer' has the highest rank and 'contact' has the third rank in TF/IDF ranking.

TermRank retrieves each node and edge in each iteration. That means the total running time for k iterations is $O\left(k(n+m)\right)$ utilizing hash tables for nodes and their edges. In our experiments we found that k is a small constant. Even for 322 million links, the PageRank algorithm converges in about 52 iterations [11].

5. EXPERIMENTS

Keyword	*# of pages*	*# of terms*	*Categories*
apple	648	24455	Computers(463), Fruit(136), Music(21), Locations(17), Movies(6), Games(5)
dell	33	4141	Arts(18), Computers(11), Authors(4)
gold	670	16043	Shopping(471), Mining(151), Movies(28), Motors(11), Games(8), Sports(1)
jaguar	138	8460	Cars(78), Video games(48), Animals(9), Music(3)
jordan	444	15122	Country(249), Music(42), Authors(38), City(29), Basketball(10), Genealogy(8), Banks(10), Movies(10), Soccer(1), News(4)
saturn	71	5004	Cars(22), Planets(21), Anime(19), Video games(9)
tiger	108	8017	Sports(35), Video games(28), Animals(25), Movies(19), Terrorism(3)

Table 2: Selected search keywords and their corresponding categories in data sets.

In our experiments, we identified some ambiguous keywords inspired from [16] on *Open Directory Project*[4] (ODP). ODP is the largest, most comprehensive human-edited directory of the Web. It is constructed and maintained by a vast, global community of volunteer editors. We selected the search keywords 'apple', 'dell', 'gold', 'jaguar', 'jordan', 'saturn' and 'tiger' for data collection comprising 2,112 Web pages and 81,422 unique terms in total. These document collection statistics are presented in Table 2. During the preprocessing step, common data types such as percentages, dates, numbers etc., stop words, and punctuation symbols are filtered using simple regular expressions.

We used two clustering algorithms to demonstrate the efficacy of TermRank: K-means and SCuBA [1]. K-means is one of the most common clustering methods preferred for its speed and quality. In our experiments the actual number of clusters K is provided according to the number of matching ODP categories. For each keyword data, K-means has been executed 20 times and the results correspond to the average of all runs. Purity and F-measure deviate in the interval of $\pm 5\%$. SCuBA is a state-of-art subspace clustering algorithm that efficiently determines clusters and their related features (subspaces) by analyzing frequent term sets of documents. It is originally part of an article recommendation system for researchers.

The experiments run on an Intel Pentium4 3GHz CPU with 1GB RAM which had Windows XP operating system.

[4] `http://www.dmoz.org`

Keyword	*K-Means*									*SCuBA*								
	TF			TF/IDF			TermRank			TF			TF/IDF			TermRank		
	P	E	F	P	E	F	P	E	F	P	E	F	P	E	F	P	E	F
apple	0.81	0.84	0.42	0.86	0.86	0.46	0.90	0.57	0.56	0.85	0.65	0.45	0.84	0.69	0.43	0.87	0.60	0.47
dell	0.80	0.65	0.29	0.80	0.65	0.26	0.90	0.28	0.36	1.00	0.00	0.54	1.00	0.00	0.55	1.00	0.00	0.58
gold	0.78	0.65	0.40	0.74	0.87	0.43	0.84	0.74	0.53	0.71	1.11	0.42	0.73	1.03	0.40	0.76	0.95	0.46
jaguar	0.86	0.71	0.41	0.89	0.61	0.44	0.91	0.50	0.52	0.95	0.14	0.19	0.92	0.19	0.19	1.00	0.00	0.24
jordan	0.49	2.51	0.30	0.52	2.10	0.33	0.62	1.76	0.46	0.77	1.04	0.36	0.76	1.14	0.38	0.80	1.01	0.42
saturn	0.51	1.06	0.45	0.55	1.46	0.46	0.76	0.95	0.63	0.68	0.82	0.31	0.71	0.78	0.27	0.80	0.51	0.40
tiger	0.56	1.37	0.48	0.51	1.35	0.45	0.63	1.25	0.54	0.78	0.61	0.46	0.84	0.47	0.51	0.86	0.43	0.58
overall	0.73	1.07	0.39	0.74	1.00	0.42	0.83	0.77	0.53	0.79	0.84	0.40	0.79	0.84	0.40	0.83	0.74	0.45

Table 3: Performance comparison of term ranking methods in K-means and SCuBA. *P*, *E* and *F* refers to purity, entropy and F-measure respectively.

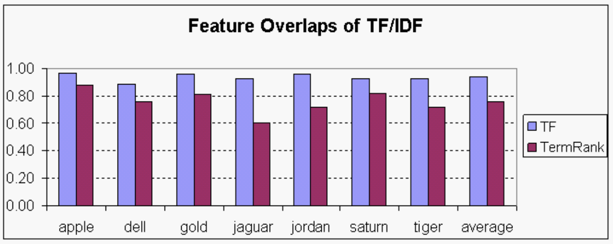

Figure 3: The overlap of TF/IDF features with the other methods.

Keyword	*Sample features*	*Rel. Category*
apple	cake, pudding, fruit, recipe, pie, bread	Fruit
dell	computer, desktop, business, software	Computers
gold	exploration, mineral, mining, resources, panning, company	Mining
jaguar	classic, british, club, car	Cars
jordan	national, foreign, middle, east	Country
saturn	playstation, game, arcade	Video games
tiger	movie, martial, art, hidden, dragon, crouching, chinese	Movies

Table 4: Sample subspaces and their related categories. One subspace is shown for each search keyword.

5.1 Evaluation Metrics

To evaluate the quality of our results, the computed clusters are compared with the actual categories given in ODP. We use the common evaluation metrics for clustering [12]: *precision*, *recall*, *F-measure*, *purity*, and *entropy*. Precision, $p_{ij} = \frac{n_{ij}}{n_i}$ and recall, $r_{ij} = \frac{n_{ij}}{n_j}$ compare each cluster i with each category j where n_{ij} is the number of Web pages appear in both the cluster i and the category j, n_i and n_j are the number of Web pages in the cluster i and in the category j respectively. F-measure, $F_{ij} = \frac{2p_{ij}r_{ij}}{p_{ij}+r_{ij}}$ is a common metric calculated similarly to the one in IR. The F-measure of a category j is $F_j = \max_i\{F_{ij}\}$ and similarly the overall F-measure is:

$$F = \sum_j \frac{n_j}{n} F_j. \qquad (4)$$

Quality of each cluster can be calculated by purity and entropy. Purity measures how pure is the cluster i by $\rho_i = \max_j\{p_{ij}\}$. The purity of the entire clustering can be calculated by weighting each cluster proportional to its size as:

$$\rho = \sum_i \frac{n_i}{n} \rho_i \qquad (5)$$

where n is the total number of Web pages.

The entropy of a cluster i is $E_i = -\sum_j p_{ij} \log p_{ij}$. Calculating the weighted average over all clusters gives the entire entropy of the clustering:

$$E = \sum_i \frac{n_i}{n} E_i. \qquad (6)$$

5.2 ODP Results

First we use a wrapper which sends a given search keyword to `http://www.dmoz.org`, and collects the resulting categories and the Web pages that belong to those categories. Collected pages are categorized by their ODP categories. Next, all blocks are extracted from the collected Web pages. About 5% of the collected Web pages in ODP are defective such as incorrect or redirected urls, erroneous HTML codes, pages composed of just images, or under construction. For all Web search result collections, TermRank is quite fast and it runs in less than a milisecond.

The term overlaps in the top-200 for TF/IDF vs. TF and TermRank are given in Figure 3. TF/IDF and TF have significant overlap – about 94% on the average. Whereas TermRank has overlap of only 76%. Hence it is shown TF and TF/IDF perform very similarly (almost in ±3% range) as detailed in Table 3. Overall performances of TF and TF/IDF for both K-means and SCuBA are very close. On the other hand, TermRank have performed substantially better for both K-means (purity $\geq +9\%$ and F-measure $\geq +11\%$) and SCuBA (purity $\geq +4\%$ and F-measure $\geq +5\%$).

The overlaps in the terms generated by TermRank are the lowest in *jaguar* and *jordan* data sets as shown in Figure 3. Discriminating terms of the categories are well identified and properly ranked hence one can see the significant difference between F-measures of TF, TF/IDF and TermRank in Table 3.

TermRank is shown to be successful in degrading the common terms. Table 5 presents some obvious common terms and their ranks. Common terms are indeed placed at much lower ranks by TermRank than by TF/IDF. Especially in *dell*, *jordan*, *saturn* and *tiger* data sets, TermRank very successfully reduces the ranks of the common terms. That's why TermRank performs better than the other methods in these data sets.

The quality of clustering in a document collection is affected by other factors such as *context dominance* and *context overlap*. If most of the Web pages belong to one category, the context of that category dominates the others. As a consequence, a large amount of the feature terms are generated from the context of the dominating category and their ranks are high. In that case, clustering quality reduces since the there is not sufficient terms to distinguish remaining small categories. *Apple*, *gold* and *jordan* are such examples in our data sets. *Computer*, *Shopping* and *Country* are the dominating categories in these data sets respectively. In the second case, Web pages in different categories have

	apple		dell		gold		jaguar		jordan		saturn		tiger	
	tfidf	tr	tfidf	tr	tfidf	tr	tfidf	tr	tfidf	tr	tfidf	tr	tfidf	tr
search	465	581	58	411	147	257	181	870	>1000	>1000	141	393	428	>1000
contact	377	557	49	49	128	218	110	192	344	730	192	>1000	374	873
links	110	276	212	533	91	141	35	179	100	561	42	184	170	195
email	151	184	206	>1000	166	227	218	556	639	>1000	898	>1000	218	>1000
news	73	198	199	>1000	75	217	22	283	20	433	186	960	64	231
online	23	142	587	>1000	43	171	175	157	305	>1000	262	499	173	>1000
special	47	537	431	>1000	85	173	246	246	203	297	513	>1000	144	300
copyright	34	202	45	271	198	560	278	960	23	166	>1000	>1000	>1000	>1000
rights	57	124	>1000	>1000	285	690	812	>1000	>1000	>1000	>1000	>1000	>1000	>1000
top-200	7	4	4	1	8	3	5	3	3	1	4	1	4	1

Table 5: Ranks of some common terms in different data sets. 'tfidf' and 'tf' refers to TF/IDF and TermRank respectively. 'top-200' row specifies the number terms ranked in top-200.

similar contexts. This is not due to the context ambiguity or the ambiguity of the terms in the context but exactly the same context might appear in more than one categories. For instance, many Web pages in *Shopping* category have information on 'gold' as a material which is common in Mining. Another example is *Sports* and *Video games* categories in *tiger* data set. 'Tiger Woods' is the well-known famous golf player and 'golf' is the main context of the 'Tiger Woods' Web pages in *Sports* category. However, the context is again 'golf' in the 'Tiger Woods' video game pages in *Video Games* category. That is the main reason of the purity of *tiger* data set to be lower as presented in Table 3.

High quality term features generated by TermRank can be very useful in subspace clustering algorithms to produce precise subspaces. To demonstrate the efficacy, some of the subspaces generated by SCuBA are presented in Table 4. Subspaces that serve as the contextual terms of the categories are accurately identified from the terms generated by TermRank. For example, in *tiger* data set the subspace refers to the Web pages of 'Crouching Tiger, Hidden Dragon', the famous movie with 4 Oscar awards in 2001. The movie is originally 'Chinese' and about 'martial arts'.

6. CONCLUSION

In this paper, we show the ineffectiveness of frequency based term ranking methods such as TF and TF/IDF for clustering the Web search results. Instead, we provide a novel term ranking method, TermRank which utilizes random walks on a relational graph of the given Web page collection. Our experimental results illustrate the effectiveness of our algorithm by measuring purity, entropy and F-measure of generated clusters based on Open Directory Project (ODP) data.

In this work, we ranked the terms without explicitly categorizing them into the three categories; discriminative, ambiguous and common terms. Future work includes clearly separation of terms into these categories. By categorizing the terms, common terms can be easily excluded whereas discriminative and ambiguous terms can be more effectively used in clustering Web search results.

7. REFERENCES

[1] N. Agarwal, E. Haque, H. Liu, and L. Parsons. A subspace clustering framework for research group collaboration. *International Journal of Information Technology and Web Engineering*, 1(1):35–38, 2006.

[2] F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *SIGKDD*, pages 436–442, New York, NY, USA, 2002. ACM Press.

[3] D. Cheng, S. Vempala, R. Kannan, and G. Wang. A divide-and-merge methodology for clustering. In *PODS*, pages 196–205, New York, NY, USA, 2005. ACM Press.

[4] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *WWW*, pages 801–810, New York, NY, USA, 2005. ACM Press.

[5] E. Gabrilovich and S. Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of The Nineteenth International Joint Conference for Artificial Intelligence*, pages 1048–1053, Edinburgh, Scotland, 2005.

[6] M. Gori and A. Pucci. Itemrank: A random-walk based scoring algorithm for recommender engines. In *IJCAI*, pages 2766–2771, 2007.

[7] S. Huang, Z. Chen, Y. Yu, and W.-Y. Ma. Multitype features coselection for web document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 18(4):448–459, 2006.

[8] A. Leouski and W. B. Croft. An evaluation of techniques for clustering search results. Technical Report IR-76, University of Massachusetts, Amherst, 1996.

[9] T. Liu, S. Liu, Z. Chen, and W.-Y. Ma. An evaluation on feature selection for text clustering. In *ICML*, pages 488–495, 2003.

[10] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[12] M. Rosell, V. Kann, and J.-E. Litton. Comparing comparisons: Document clustering evaluation using two manual classifications. In *ICON*, 2004.

[13] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[14] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *AAAI*, pages 58–64. AAAI, July 2000.

[15] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to Web search results. *Computer Networks*, 31(11–16):1361–1374, 1999.

[16] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *ACM SIGIR*, pages 210–217, New York, NY, USA, 2004. ACM Press.