Efficient Jaccard-based Diversity Analysis of Large Document Collections

Fan Deng
L3S Research Center*
Appelstr.9a
Hannover, Germany
deng@L3S.de

Stefan Siersdorfer
L3S Research Center
Appelstr.9a
Hannover, Germany
siersdorfer@L3S.de

Sergej Zerr
L3S Research Center
Appelstr.9a
Hannover, Germany
zerr@L3S.de

ABSTRACT

We propose two efficient algorithms for exploring topic diversity in large document corpora such as user generated content on the social web, bibliographic data, or other web repositories. Analyzing diversity is useful for obtaining insights into knowledge evolution, trends, periodicities, and topic heterogeneity of such collections. Calculating diversity statistics requires averaging over the similarity of all object pairs, which, for large corpora, is prohibitive from a computational point of view. Our proposed algorithms overcome the quadratic complexity of the average pair-wise similarity computation, and allow for constant time (depending on dataset properties) or linear time approximation with probabilistic guarantees. Our theoretical findings are verified on synthetic and real-world data sets. As applications, we show examples of diversity-based studies on large samples from corpora such as the social photo sharing site Flickr, the DBLP bibliography, and US Census data.

Categories and Subject Descriptors

H.3.0 [**Information Storage and Retrieval**]: General

Keywords

diversity, efficiency, min-wise hashing, random sampling

1. INTRODUCTION

Diversity has been studied in different disciplines and contexts for decades. The diversity of a population can reveal certain cultural properties of a country [18, 11], e.g. with respect to religion, ethnic groups, or political orientations. In the area of ecology, biodiversity is used as a measure of the health of biological systems [15]. Recently, diversity also drew the attention of scientists in the database and information retrieval communities. For instance, Vee et al. [29] use the sum of similarities of all object pairs to measure diversities of relational records, while Ziegler et al. [30] employ a similar measure for diversifying items in a recommender system. In the area of search result diversification [12, 30], inter-object similarity is used to obtain a subset of the most dissimilar results, providing an overview over the result space. However, due to the quadratic computational complexity, the mentioned approaches are applied on relatively small sets, limited to the number of life forms in a biosystem, or top-k objects selected in the context of search result diversification. In contrast, in this paper, we focus on analyzing diversity on Web collections and in other large-scale text corpora *as a whole*.

Motivation: Increasing amounts of data are published on the Internet on a daily basis, not least due to popular social web environments such as YouTube, Flickr, and the blogosphere. This results in a broad spectrum of topics, communities and knowledge, which is constantly changing over time. An increase of content diversity over time indicates that a community is broadening its area of interest; negative peaks in diversity can additionally reveal a temporary focus on specific events. Analyzing diversity is promising for understanding the dynamics of information needs and interests of the users generating the data. In a recommender system context, diversity and its temporal evaluation exhibits an additional criterion for suggesting to users interest groups or communities (e.g. rather broad vs. more specialized ones). Additionally, our methods could be used to efficiently analyze the correlation between diversity indexes for the documents retrieved for a set of queries and the performance of an IR system, allowing for quicker, deeper, and broader analyses. Furthermore, high diversity in document repositories can be employed as indicator that more structure is required, and trigger manual or automatic processes for introducing topic hierarchies or clusters (cf. Section 6.3). In our studies we show an example analysis depicting the temporal development of the diversity of photo annotations in Flickr over time, revealing interesting insights about trends and periodicities in that social content sharing environment (Figure 1 in Section 6). In addition we conduct an analysis of scientific communities on data extracted from the DBLP bibliography, and furthermore, study diversity in clustered corpora such as US Census data and newsfeeds. Our goal is to enable a fast analysis of the variation of topic diversities according to different aspects such as time, location and communities.

Computational Problem: There are a number of well-known indexes measuring diversity in ecology such as Simpson's diversity [27] and the Shannon index [14]. For applying the concept of diversity to *text data*, existing works use the sum of all document pair similarities or variations based on pair-wise distances as diversity metrics [24, 12, 30, 29]. These diversity indexes are based on the computation of pairwise comparisons. Thus, a common problem is their computational complexity: $O\left(n^2\right)$ comparisons are necessary for sets of n objects. While this is still feasible in scenarios with small amounts of data as for top-k candidates in query result diversification, it becomes prohibitive when computing topic diversity for large data sets.

[1]until 2011

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

Contribution: In order to solve the computational problem (the main contribution of this paper), we propose two novel algorithms which make use of random sampling and Min-wise independent hashing paradigms. More specifically, we propose two fast approximation algorithms for computing the average pair-wise Jaccard similarity of n sets with probabilistic accuracy guarantees, coined as *SampleDJ* and *TrackDJ*. We discovered that specific properties of the Jaccard coefficient as underlying measure for the pair-wise similarities required in diversity indexes can make the computation feasible, even for huge amounts of data. Although there exist a variety of alternative metrics, Jaccard is still one of the most popular measures in IR due to its simplicity and high applicability [19, 3], and provides intuitive and interesting results in our example studies. SampleDJ, which is based on random sampling, solves the problem in $o\left(\frac{1}{\mathrm{RDJ}^2}\right)$ time independent of the data set size, where RDJ is the diversity index value to be estimated. TrackDJ, which is based on Min-wise independent hashing [7, 4, 5, 6] solves the problem in $O(n)$ time regardless of the input data distribution. Experiments on real-world as well as synthetic data confirm our analytical results. Furthermore, we show the applicability of our methods in example studies on various data collections.

Outline: The rest of this paper is organized as follows: In Section 2, we describe related work in the areas of diversity measurement, diversification of search results, sampling, and Min-wise hashing. Section 3 introduces the diversity index used throughout this paper. In our main Section 4, we describe algorithms for efficiently computing the diversity index values, and prove accuracy and efficiency properties. We verify our theoretical findings using real and synthetic data in Section 5. As application examples, we show results of various corpora studies in Section 6. Finally, we conclude and point out directions for future work in Section 7.

2. RELATED WORK

Diversity measurement: Diversity indexes were proposed in ecology [27, 14] decades ago; they assume that objects are classified into groups. For example, Simpson's diversity index [27], a well-known diversity index in ecology is defined as follows: $D = \sum_{i=1}^{Z} \pi_i^2$, where π_i is the fraction of individuals belonging to group i and each individual in the population belongs to one of Z groups. Note that Simpson's index can just be employed in the special case where objects belong to one of a discrete set of categories. Recently, new diversity indexes were proposed in various areas focusing on different applications. For example, Stirling [28, 25] proposed a general diversity index trying to measure diversities in different areas in science, technology and society. The general Stirling index is defined as $D = \sum_{ij(i \neq j)} (d_{ij})^{\alpha} (p_i p_j)^{\beta}$, where d_{ij} is the distance between object i and j; p_i, p_j are the relative occurrences of object i and j in the whole data set. The diversity measure used in this paper for text collections can be interpreted as a special case of the Stirling index. *None of the mentioned works deals with the efficiency problem of computing diversity measures for whole data collections, which is the main focus of this paper.*

Diversity in search or query result diversification: There is a plethora of work on diversifying query or search results. Search engines, recommender systems and databases have the need to return diverse results to users in order to include answers for different user needs in the result set. This is especially important if the query is ambiguous. Vee et al. [29] use the sum of similarities of all object pairs to measure diversities of relational records, while Ziegler et al. [30] apply a similar measure for diversifying items in a recommender system. Gollapudi and Sharma [12] make use of inter-object similarity in their axiomatic search result diversification approach. In their recent work [21] Minack et al. deal with efficiently diversifying search results for large data streams; note that obtaining a diverse *top-k list* from a large dataset corresponds to an entirely different problem setting than the one studied in this paper. *In general, work on search result diversification is about diversifying small top-k sets of query results. In contrast, in this paper we tackle the efficiency problem of computing diversity indexes for whole data collections.*

Sampling is a technique that finds a wide range of applications, such as auditing of large databases and query optimization in DBMS [22]. A more theoretical example related to our work is the application of random sampling to estimate the average value (or sum) of a set of numbers, for which it is known to be difficult to bound the relative error without having prior knowledge about the input data [9]. More research on sampling can be found in the survey of Babcock et al. [2] and in [22]. *We are the first to use random sampling as an underlying technique to solve the problem of efficiently computing diversity indexes.*

Min-wise independent hashing is a technique originally proposed for finding near duplicate documents [6]. It can be considered as a variation of Locality Sensitive Hashing (LSH) [13, 10]. LSH was used for indexing high dimensional data points. Similar to Min-wise independent hashing, LSH also has the property that two objects with a smaller distance are more likely to have a hash collision. The difference is that the distance/similarity is measured by the L_p norm (e.g. Euclidean or Hamming distance of vectors). For cosine similarities of vectors, there exist also hash functions with the LSH property [8]. *Different from the aforementioned applications, we are the first to use Min-wise independent hashing as a building block for efficient diversity index estimation.*

To the best of our knowledge, there exist no other methods for efficiently computing the diversity of whole corpora.

3. DIVERSITY INDEX DEFINITION

There exists a plethora of methods for computing pair-wise similarities (e.g. cosine similarity, Jaccard coefficient, Okapi BM25, inverted distances, etc.). For diversity computation in this paper, we employ the Jaccard coefficient, which is one of the most popular measures in IR due to its simplicity and high applicability [19, 3]. We will see that specific properties of this coefficient can make the computation of diversity values feasible even for large data sets.

Given two term sets O_i and O_j (e.g. sets of tags for two photos in Flickr), their Jaccard similarity $\mathrm{JS}(O_i, O_j)$ is defined as follows:

$$\mathrm{JS}(O_i, O_j) = \frac{|O_i \cap O_j|}{|O_i \cup O_j|}.$$

We define our topic diversity index as *Refined Diversity Jaccard-Index*, which is in fact the average Jaccard similarity of all object pairs.

DEFINITION 1 (REFINED DJ-INDEX). *Given a collection of objects $O_1, \ldots, O_n$, where each object is a set of elements (e.g. terms), the refined DJ-Index measuring the diversity of the collection is defined as follows:*

$$\mathrm{RDJ} = \frac{2}{n(n-1)} \sum_{i<j} \mathrm{JS}(O_i, O_j),$$

where $1 \leq i < j \leq n$.

In this paper, we use the expression *Refined* DJ-Index, or *RDJ index* for short, to emphasize that self-similar pairs are not included in the diversity computation. It is easy to see that the RDJ-index can be considered as a special case of the Stirling index described in Section 2 where α and β are set to 1, and both p_i and p_j are $1/n$ if self-similar pairs are included. Note that smaller RDJ values mean larger diversities. For better visualization in plots we do not use 1-RDJ to measure diversity because RDJ values are usually quite small, and 1-RDJ is often close to 1.

4. DIVERSITY INDEX COMPUTATION

Having defined the RDJ-index for measuring diversity, the goal of this section is to compute this statistic for a given text corpus.

PROBLEM 1 (RDJ-INDEX COMPUTATION). *Given a collection of objects $O_1, \ldots, O_n$, where each object is a set of elements (e.g. terms), the objective is to compute the RDJ-index value efficiently.*

4.1 The Naive Method: All-Pair

The straightforward solution for computing RDJ directly according to Definition 1 is 1) to compare all object pairs and compute the size of intersection and union of each pair, and then 2) to sum up the similarities of all pairs and to divide by the number of object pairs. We call this algorithm *All-Pair*. Clearly, All-Pair takes $O\left(n^2\right)$ time, and is inefficient for data collections with a large number of objects. Although some heuristics may improve the efficiency of All-Pair, the time complexity will still be $O\left(n^2\right)$ since in the worst case all pairs will have to be compared to obtain the exact value. Note that All-Pair can be easily parallelized with $O\left(n^2/m\right)$ comparisons where m is the number of machines, which, however, is still not feasible for large datasets. In the following, we study approximate but more efficient techniques to estimate the value of the RDJ diversity index.

4.2 Diversity Index Approximation

To increase the time efficiency of the RJD computation, we consider approximation algorithms that are usually faster but provide estimated results, which are subject to errors. Our goal is to bound these errors within a tolerable range. In the following, we define error measures for the estimation quality of approximation algorithms.

DEFINITION 2 (ABSOLUTE ERROR). *Let $\hat{D}$ be an estimate of a statistic D, the absolute error is defined as follows:*

$$\left|\hat{D} - D\right|.$$

DEFINITION 3 (RELATIVE ERROR). *Let $\hat{D}$ be an estimate of a statistic D, the relative error is defined as:*

$$\frac{\left|\hat{D} - D\right|}{D}.$$

Like many other indexes, diversity indexes are mainly used for comparisons. A single index value alone usually cannot reveal much insight to users, and the relative error is more appropriate than the absolute error for measuring the accuracy of index value estimates. For the absolute error, it becomes hard for users to specify a meaningful accuracy requirement in form of an error bound unless they approximately know the value of the indexes to be estimated. Therefore, for both approximation algorithms described in the next subsections, we will focus our theoretical analysis on the relative error.

4.3 The SampleDJ Algorithm

A natural solution to reduce the computational cost is sampling. One approach is to take a random sample of the input data set, i.e., to sample r objects uniformly at random (without replacement) from a collection of n objects, compute the sum of similarities of all object pairs in the sample, and scale the diversity index of the sample. Another approach is to take r objects uniformly at random but with replacement (see also [22] for a general discussion of sampling with replacement). In our context, we sample from all possible *pairs*, compute the similarities of those r pairs and scale the result according to r and n as the final estimate. The advantage of this approach is that each pair is taken independently of other pairs, which makes the sampling analysis easier. We focus on this approach in this paper and name it *SampleDJ*.

Algorithm 1: SampleDJ: An approximation algorithm estimating diversity indexes

Input: A collection of objects $O_1, \ldots, O_n$, each object is a set of term IDs (assuming the term IDs in each object are sorted); relative error bound ϵ, accuracy confidence $1-\delta$, the number of consecutive trials W (e.g. 10000) before checking the accuracy and tolerable overall running time T.

Output: Estimates of RDJ-Index of the given collection

begin

1. Load the data set into memory;
2. $r_1 = 0$; $r_2 = \lceil \log_2 \frac{1}{\delta} \rceil$;

for $i = 1, \ldots, r_2$ **do**

3. jsSum[i] = 0;

repeat

for $i = 1, \ldots, r_2$ **do**

for $j = 1, \ldots, W$ **do**

4. Generate 2 distinct integers, A & B, uniformly at random within the range [1, n];
5. jsSum[i] = jsSum[i] + JS(O_A, O_B);

6. $r_1 = r_1 + W$;
7. $\widehat{\text{RDJ}} = \frac{\text{median(jsSum[1],...,jsSum[}r_2\text{])}}{r_1}$;
8. $\Delta = \sqrt{\frac{1}{r_1}}$;

until $\frac{\Delta}{\left|\widehat{\text{RDJ}} - \Delta\right|} <= \epsilon$ *or Running time* $> T$;

if $\frac{\Delta}{\left|\widehat{\text{RDJ}} - \Delta\right|} <= \epsilon$ **then**

9. Return $\widehat{\text{RDJ}}$.

else

10. Return FAIL;

The details are shown in Algorithm 1. The input to the algorithm is a collection of objects with each object consisting of a set of term IDs. In the Line 2 the values of r_1 and r_2 are initialized depending on the desired accuracy confidence; here, r_1 is the number of trials in each of the r_2 experiments. The final result is computed as the median RDJ value obtained from the r_2 independent experiments (r_2 being determined by user requirement δ); r_1 increases with each iteration of the algorithm while r_2 remains constant. In the Line 4 and 5 the similarity sum of all sampled pairs is computed. Line 6 keeps track of the number of trials. Line 7 computes the current RDJ estimate. Line 8 and the "until" statement serve to determine if the RDJ estimate is accurate enough.

Note that window size W is independent of data set sizes, and merely corresponds to regular accuracy checks in the algorithm; the number of windows is determined *automatically*, and changes according to the data properties. Therefore, no tuning of W is required. In our experiments we used the same value for W (10,000) for all of the datasets. For practical reasons, there is also a "tolerable running time" parameter T because SampleDJ can take more running time than the straightforward method in the worst case, and this time is not predictable. For example, if all object pairs have zero similarity, SampleDJ would proceed forever without a stopping condition based on T. The algorithm stops when the estimated result is accurate enough or the running time exceeds the user-specified time limit. We name the former case as "fail" and the latter as "succeed". Based on our analysis described in the next paragraph, the accuracy is determined by the absolute error bound

Δ, which can be determined based on the number of trials r_1. Note that ϵ and δ are the user requirements for the accuracy of the diversity index computation, and r_2 is determined by user input δ.

Since SampleDJ is an approximation algorithm, it is of high practical importance to have accuracy guarantees. The next statement specifies the time and space complexity of the SampleDJ algorithm.

CLAIM 1 (COMPLEXITY OF SAMPLEDJ). *If Algorithm 1 succeeds, the estimate $\widehat{\mathrm{RDJ}}$ is guaranteed to have a relative error at most ϵ with probability at least $1-\delta$. That is,*

$$\Pr\left[\frac{|\widehat{\mathrm{RDJ}} - \mathrm{RDJ}|}{\mathrm{RDJ}} > \epsilon\right] < \delta,$$

where RDJ *is the true value of $\widehat{\mathrm{RDJ}}$. The time and space costs to guarantee a success are $O\left(\frac{U\cdot\log_2 1/\delta}{\epsilon^2\,\mathrm{RDJ}^2}\right)$ and $O(nU)$ respectively, where U is the maximum set size of all objects.*

PROOF. We first prove the accuracy of SampleDJ. Let $p_1,\ldots,p_N$ be all possible object pairs, where $N=\frac{n(n-1)}{2}$. Let X be a random variable indicating the Jaccard similarity of a pair drawn uniformly at random from $p_1,\ldots,p_N$. That is, $X=s_i$ $(i=1,\ldots,N)$ with $\Pr[X=s_i]=\frac{1}{N}$, where s_i is the Jaccard similarity of p_i. Then $E[X]=\frac{1}{N}\sum_{i=1}^{N}s_i=\mathrm{RDJ}$, and

$$\begin{aligned}\mathrm{VAR}[X] &= E\left[X^2\right]-(E[X])^2=\frac{1}{N}\sum_{i=1}^{N}s_i^2-\mathrm{RDJ}^2\\ &\le \mathrm{RDJ}(1-\mathrm{RDJ}).\end{aligned}$$

If there are r_1 identically and independently distributed random variables following the same distribution as X, the mean of the r_1 variables is also expected to be equal to RDJ and the variance of the mean is at most $\frac{\mathrm{RDJ}(1-\mathrm{RDJ})}{r_1}$. According to Chebyshev's Inequality, we have

$$\Pr\left[\frac{\left|\widehat{\mathrm{RDJ}}-\mathrm{RDJ}\right|}{\mathrm{RDJ}}>\epsilon\right] < \frac{\mathrm{RDJ}(1-\mathrm{RDJ})}{r_1\epsilon^2\,\mathrm{RDJ}^2} \le \frac{1}{4r_1\Delta^2},$$

where absolute error bound $\Delta=\epsilon\cdot\mathrm{RDJ}$. Let the probability bound $\frac{1}{4r_1\Delta^2}=\frac{1}{4}$, then $\Delta=\sqrt{\frac{1}{r_1}}$. By repeating the same experiment $\log_2\frac{1}{\delta}$ times independently and taking the median, the upper bound of the probability $\Pr\left[\left|\widehat{\mathrm{RDJ}}-\mathrm{RDJ}\right|>\epsilon\,\mathrm{RDJ}\right]$ is raised to $\frac{1}{4}^{\frac{\log_2\frac{1}{\delta}}{2}}=\delta$.

Next, we show the time and space complexities. To guarantee the relative error at most ϵ with probability $1-\delta$, it is sufficient to guarantee the absolute error at most Δ since $\left|\widehat{\mathrm{RDJ}}-\Delta\right|\le\mathrm{RDJ}$, $\frac{\Delta}{\left|\widehat{\mathrm{RDJ}}-\Delta\right|}\ge\frac{\Delta}{\mathrm{RDJ}}=\epsilon$. Therefore, setting $r_1=\frac{1}{\Delta^2}=\frac{1}{\epsilon^2\,\mathrm{RDJ}^2}$ guarantees the relative error with probability $1-\delta$ according to the accuracy analysis. Thus, the time complexity is $O\left(\frac{U\cdot\log_2 1/\delta}{\epsilon^2\,\mathrm{RDJ}^2}\right)$. As for the space complexity, the main costs come from storing the input, which is $O(nU)$. □

Note that the running time of SampleDJ is independent of n, but dependent on RDJ, the value to be estimated. Thus, SampleDJ can be very efficient for large data sets if RDJ is reasonably large. However, for smaller values of RDJ, the running time can be higher. Note further that, similar to All-Pair, SampleDJ can be easily parallelized; this is because the pairwise similarity computations, which correspond to the main computational overhead, and part of the aggregation steps involved can be run independently on different machines.

4.4 TrackDJ: Estimating the RDJ-Index in Linear Time

Although SampleDJ can be very efficient in some cases, its performance is sensitive to the data distribution, and it can be very slow in the worst case. Apart from that, without prior knowledge of the input data, it is difficult to predict the running time. In this section we present TrackDJ, another approximation technique returning an accurate estimate for the DJ-Index in $O(n)$ time regardless of the input data distribution, where n is the number of objects (term sets) in the data set. We first briefly sketch the underlying concept of Min-wise hashing and its application in our scenario.

4.4.1 Prerequisites: Min-wise Independent Hashing

Broder et al. proposed a powerful technique called *Min-wise independent hashing* (Min-hash) [7, 4, 5, 6]. An interesting property of this technique is that the hashing collision probability of two objects is exactly equal to their Jaccard similarity. The basic idea of Min-hash mapping a term set is as follows: Min-hash picks a permutation π uniformly at random from all possible permutations of the term vocabulary. Given a set of terms O, Min-hash applies π on O and takes the term that has the minimum rank after the permutation as the Min-hash value.

EXAMPLE 1 (MIN-WISE INDEPENDENT HASHING). *Given 3 objects $O_1(A,B,C)$, $O_2(B,C,D)$ and $O_3(C,E)$ where A, B, C, D, E are distinct terms. Assume TrackDJ uses two min-hash functions h_1 and h_2 where the two permutations are: from (A,B,C,D,E) to $\pi_1=(E,B,A,C,D)$ and $\pi_2=(A,C,B,D,E)$. Under these permutations, the terms with minimum ranks are as follows:*

$$h_1(O_1)=B, h_1(O_2)=B, h_1(O_3)=E,$$
$$h_2(O_1)=A, h_2(O_2)=C, h_2(O_3)=C.$$

4.4.2 The TrackDJ Algorithm

Given the Min-wise hashing property that more similar objects are more likely to have a hash collision, TrackDJ counts the number of collisions of all object pairs and estimates the diversity index. This is done by finding the self-join sizes[2] of all min-hash values. The self-join size of a set of items is in fact the similarity sum of all object pairs where the similarity function returns only 1 or 0 depending on whether the pair of objects match or not. In summary, the key idea of TrackDJ is that if there are more similar pairs, the self-join size of the min-hash values will be higher due to the Min-wise hashing property; instead of comparing all object pairs, the self-join size of a set of items can be computed in linear time.

Using the above Min-wise Hashing example, the self-join sizes of min-hash values under h_1 and h_2 are both 5. Accordingly, TrackDJ returns $\frac{5-3}{3(3-1)}$ as the RDJ-Index estimate where the 5 in the numerator is the average of the two self-join sizes, and the 3 in the denominator is the number of objects. In our example, the estimate is exactly the true RDJ-Index value. Although this may not always be the case in practice, the *expected* value of the estimate is equal to the true value, which we will prove below. Typically, TrackDJ needs to use a larger number of min-hash functions to reduce the variance.

TrackDJ maps each object (e.g. a term set) to a min-hash value. The algorithm uses the fact that two objects have a higher probability to have a min-hash collision if they have a higher Jaccard similarity; the total number of collisions of all possible object pairs

[2]The Self-join size of a set of items, F, is defined as $F=\sum_i f_i^2$, where f_i is the number of occurrences (frequency) of item i in the data set.

Algorithm 2: TrackDJ: Estimating the RDJ-Index in Linear Time

Input: A collection of objects $O_1, \ldots, O_n$, each object is a set of IDs of terms ranging from 0 to $D-1$; a user specified relative error bound ϵ and confidence $1-\delta$; maximum set size U among all objects

Output: an estimate of RDJ-Index of the given collection

begin

1. $L_1 = \left\lceil \frac{8(U-1)}{\epsilon^2} \right\rceil$; $L_2 = \lceil \log_2 \frac{1}{\delta} \rceil$;
2. Initialize counters $f[j]$ storing L_2 self-join sizes of min-hash values;
for $l = 1, \ldots, L_2$ **do**
   3. $f[l] = 0$; //to store self-join sizes
4. Initialize counters $\mathrm{IdCounts}[]$ storing the frequency of each min-hash values;
for $l = 1, \ldots, D$ **do**
   // (D is the number of distinct terms in the data set)
   5. $\mathrm{IdCounts}[l] = 0$; //counts of min-hash values
for $k = 1, \ldots, L_1 \cdot L_2$ **do**
   6. Pick a min-hash function $h_k[]$ uniformly at random from a min-hash function family;
for $j = 1, \ldots, L_2$ **do**
   for $l = 0, \ldots, L_1$ **do**
      for $i = 1, \ldots, n$ **do**
         7. $\mathrm{IdCounts}[h_{(j-1)L_1+l}[O_i]] + +$;
      8. $\mathrm{sj} = \sum_{\mathrm{ID}} \mathrm{IdCounts}[\mathrm{ID}]^2$; //efficiency can be improved, see the algorithm description
      9. $f[j] = f[j] + \mathrm{sj}$;
      10. Reset non-zero counters of $\mathrm{IdCounts}[]$;
11. Determine the median of $f[j], j = 1, \ldots, L_2$ and set it to fm;
12. return $\frac{(\mathrm{fm}/L_1)-n}{n(n-1)}$;

directly approximates the sum of their pair-wise similarities. Thus, by computing the self-join size of all min-hash values (i.e. the total number of collisions of all pairs), TrackDJ estimates the RDJ-Index value.

The detailed approximation method is shown in Algorithm 2. In addition to the input data set and accuracy and confidence requirements ϵ and $1-\delta$, the user needs to specify the maximum term set size among the n objects. Line 1 sets the number of independent trials and experiments. For each of the L_2 iterations there are L_1 trials. L_1 and L_2 are computed based on the accuracy and confidence requirements defined by the user. Note that for TrackDJ the number of required iterations are known in advance; thus no checks of error bounds are necessary at later stages. Details can be found in the algorithm analysis part below.

Line 2 and 3 initialize a buffer storing the L_2 self-join sizes. Line 4 and 5 initialize counters, which are used later to store the number of occurrences of each ID. In each of the L_2 iterations, there is a data stream with $L_1 \cdot n$ objects. Line 6 generates the $L_1 \cdot L_2$ min-hash functions. Line 7 increments the corresponding counter based on the min-hash value of object O_i. Line 8 and line 9 compute the accumulated self-join size so far. Note that by maintaining a linked list of non-zero counters, TrackDJ does not need to check all D counters which can significantly improve the efficiency if D is large. Line 10 resets non-zero counters of $\mathrm{IdCounts}[]$. Line 11 computes the median, and Line 12 returns the final result; it computes the average self-join sizes over L_1 iterations and removes the contribution from self-similar pairs.

4.4.3 Analysis of TrackDJ

Similar to SampleDJ, TrackDJ is a randomized algorithm which approximates the RDJ-Index. In the following, we prove complexity and accuracy guarantees for TrackDJ.

CLAIM 2. *The refined DJ-Index estimate from TrackDJ, $\widehat{\mathrm{RDJ}}$, is expected to be equal to the true value* RDJ. *The variance of $\frac{\widehat{\mathrm{RDJ}}}{\mathrm{RDJ}}$ is at most*

$$\mathrm{VAR}\left[\frac{\widehat{\mathrm{RDJ}}}{\mathrm{RDJ}}\right] \leq \frac{2(U-1)}{K},$$

where U is the maximum set size of all objects in the input object collection, and K is the number of min-hash functions used.

PROOF. We first analyze the case with only one hash function; the case with K hash functions corresponds to repeating the same experiment K times independently, which we will discuss at the end of the proof.

Let X_{ij} be a random variable indicating if a pair of objects O_i and O_j are mapped to the same min-hash value, i.e. $X_{ij} = 1$ if $h(O_i) = h(O_j)$ and 0 otherwise. Also, let $Y = \sum_{i \neq j} X_{ij}$. (Note that $X_{ij} = X_{ji}$.) Thus, $\widehat{\mathrm{RDJ}} = \frac{1}{n(n-1)} Y$. Let $s_{ij} = \mathrm{JS}(O_i, O_j)$, then $\Pr[X_{ij} = 1] = s_{ij}$, and $E[X_{ij}] = s_{ij}$. Thus,

$$E[\widehat{\mathrm{RDJ}}] = \frac{1}{n(n-1)} \sum_{i \neq j} E[X_{ij}] = \frac{1}{n(n-1)} \sum_{i \neq j} s_{ij} = \mathrm{RDJ}\,.$$

$$\begin{aligned}
\mathrm{VAR}[Y] &= E\left[\left(\sum_{i \neq j} X_{ij}\right)^2\right] - \left(E\left[\sum_{i \neq j} X_{ij}\right]\right)^2 \\
&= E\left[\sum_{i \neq j} X_{ij}^2 + \sum_{i \neq j \neq k} X_{ij} X_{ik} + \sum_{i \neq j \neq k \neq l} X_{ij} X_{kl}\right] - \left(\sum_{i \neq j} s_{ij}\right)^2 \\
&\leq \sum_{i \neq j} s_{ij}(1 - s_{ij}) + \sum_{i \neq j \neq k} s_{ij}(1 - s_{ik}) + \sum_{i \neq j \neq k \neq l} s_{ij}(1 - s_{kl})
\end{aligned}$$

Due to the definition of Jaccard similarity (sets overlap size divided by sets union size), if $s_{ij} > 0$ we have

$$s_{ij} \geq \frac{1}{2U-1}.$$

$$\therefore (1 - s_{ij}) \leq \frac{2U-2}{2U-1} \leq 2(U-1)s_{ij}.$$

$$\begin{aligned}
\therefore \mathrm{VAR}[Y] &\leq 2(U-1)\left(\sum_{i \neq j} s_{ij}^2 + \sum_{i \neq j \neq k} s_{ij} s_{ik} + \sum_{i \neq j \neq k \neq l} s_{ij} s_{kl}\right) \\
&= 2(U-1)\left(\sum_{i \neq j} s_{ij}\right)^2
\end{aligned}$$

$$\therefore \mathrm{VAR}\left[\frac{\widehat{\mathrm{RDJ}}}{\mathrm{RDJ}}\right] = \frac{\mathrm{VAR}[\widehat{\mathrm{RDJ}}]}{\left(\frac{1}{n(n-1)} \sum_{i \neq j} s_{ij}\right)^2} \leq 2(U-1).$$

If TrackDJ uses K min-hash functions independently, then $\mathrm{VAR}\left[\frac{\widehat{\mathrm{RDJ}}}{\mathrm{RDJ}}\right] \leq \frac{2(U-1)}{K}$. □

With this statement, we can derive the time and space complexities of TrackDJ.

CLAIM 3 (COMPLEXITY OF TRACKDJ). *To estimate the refined DJ-Index and guarantee a small relative error (at most ϵ) with a high probability (at least $1-\delta$), TrackDJ requires $O\left(\frac{\log(1/\delta)}{\epsilon^2} nU\right)$ time and $O(\log(nU)nU)$ memory bits, where n is the number of objects in the input data set and U is the maximum set size among all objects.*

PROOF. Knowing the variance from Claim 2, by Chebyshev's Inequality we can observe that K in TrackDJ should be set to $O\left(\frac{U}{\epsilon^2}\right)$ to bound the relative error to ϵ with a constant probability (e.g. $\frac{1}{2}$). To increase the success probability, we can repeat the process described before $2\log(1/\delta)$ times independently and report the median of the outputs. In this way, the success probability can be boosted to $1-\left(\frac{1}{2}\right)^{\log(1/\delta)}$. Therefore, TrackDJ has a running time of $O\left(\frac{\log(1/\delta)}{\epsilon^2}nU\right)$. The space costs come from storing the input and ID counters are as follows: TrackDJ needs to store $O(nU)$ counters of all term IDs, which is equal to the space requirement of the input. Each frequency counter in the buffer needs $O(\log(nU)$ bits, resulting in the overall space complexity of $O(\log(nU)nU)$. □

Note that TrackDJ can be easily parallelized with a complexity of $O(n/m)$. This can be done by partitioning the data into m parts for m machines, computing self-join sizes separately for the subsets using TrackDJ, and aggregating the result on a central machine accordingly.

5. EXPERIMENTS

We evaluated the algorithms described in this paper using both synthetic and real world datasets[3]. In this section, we first elaborate on data sets used. We then compare the three algorithms, All-Pair, SampleDJ and TrackDJ, under different efficiency aspects including accuracy, time and space costs, and parameters. Note that there exist no other methods for efficiently computing the diversity of large document corpora we could compare with.

5.1 Data

5.1.1 Real Data

Our real-world data sets were obtained from Flickr[4] and DBLP. *DBLP* [17] is a Computer Science Bibliography database containing more than 1.2 million bibliographic records. The data records in DBLP are mostly contributed by human editors and are well-structured. From DBLP, we extracted 1,256,089 paper titles based on the publication year and venue. We only considered conference and journal papers and removed books and other article types. Each paper title was considered as one object for our diversity analysis. For reasons of completeness and cleanness we focused on DBLP paper titles to mine topic diversities of computer science papers.

Finally, we gathered tag assignments for 134 Mio *Flickr* photos uniformly over the time period from 01 Jan 2005 until 05 Sept 2010. From this set we selected a subset of 25 Mio photos where each of photo contained at least 3 English tags (though a dictionary check), and performed stemming.

5.1.2 Synthetic Data

We created additional synthetic data sets in order to study the performance of different algorithms on data sets with various RDJ-index values. To this end, we generated groups of objects with the property that 1) all objects within a group had the same number of terms and were pair-wise similar with the same similarity, and 2) objects in different groups had always similarity 0. By adjusting the number of groups with different sizes (i.e. the number of objects in each group), we constructed multiple data sets with different RDJ-index values. More specifically, we constructed the data sets as follows: 1) We set U, the number of term IDs in each object to 10. 2) In each group, we generated a set of common IDs shared by all objects in the group. All other IDs in the group were distinct; in this way, by controlling the common ID numbers, we set the pair-wise similarity of all pairs in each group to around 0.5. 3) By varying the number of groups G and group size Gs, we created multiple synthetic data sets with $n = G \cdot Gs = 524,288$ objects each.

[3]Source code and data sets are available at `www.L3S.de/~deng/diversity/`.

[4]`www.flickr.com`

5.2 Implementation Details

To implement the straightforward *All Pair* algorithm, we used an array of size n to store the input file, with each array element being, in turn, an array containing the term IDs of an object. To compute the RDJ-Index values, All-Pair iterates over all possible object pairs, and computes the Jaccard similarity of each pair by linearly scanning the two sets of sorted term IDs and counting the number of common terms and the total number of terms. To implement *SampleDJ*, we used the system-provided `lrand48()` function to generate random numbers within the range 1 and n. We set W, the number of consecutive trials to 10000. For *TrackDJ*, we used linear hashing to implement Min-Wise independent hashing as suggested by Broder et al. [7]. We used the same settings for error bound and confidence as for SampleDJ.

5.3 Performance Experiments

In this set of experiments, we compared the performances of the 3 algorithms All-Pair, SampleDJ and TrackDJ on real-word and synthetic data.

5.3.1 Performance Metrics

The three metrics used to measure the performance of the algorithms were running time, space (memory) requirements, and accuracy. In terms of memory, the primary costs of All-Pair and SampleDJ were almost the same: namely, storing all term IDs, requiring e.g. about 33MB for the DBLP title IDs. TrackDJ requires more space for storing the frequency counters. Depending on the data sets, this cost can be as large as the input data set in the worst case. For the DBLP data set, it is less than 1/10 of the input data. Because the space cost is relatively clear and corresponds directly to the input data set size, we focused on running time and accuracy in our experiments. The preliminary experiments with the synthetic data on the effect different parameters were conducted on a machine with 2x Xeon5320 1.86GHz processors and 16GB of memory running CentOS 5.3. For experiments on the real world data set, we used one separate core on a more powerful machine with 8x Quad-Core AMD Opteron 2.7GHz processors and 256GB of memory running the same OS. Programs were coded in C and compiled using gcc 4.40.

5.3.2 Performance Results on Real-World Datasets

Experiments for the approximation algorithms were performed with an error bound ϵ of 0.1 and a confidence value of $1-\delta$ = 0.95. Table 1 shows the running time and error values (along with the exact RDJ values computed through All-Pair). Experimental results are consistent with our theoretical findings described in Section 4.

For the naive All-Pair solution, we observe that running times are rather small for dataset sizes of 1,000 and 10,000 but, due to its quadratic behavior, quickly become infeasible for larger sets.

Our SampleDJ approach shows the best running time behavior of the three tested algorithms. Running times stay in the order of magnitude of minutes, and are almost constant with respect to the input size. The shorter running times for relatively small data sizes are an artifact of the hardware, and can be explained by L1 and L2 caching as for small datasets a larger part of possible object pairs can be kept in the caches; for larger sizes we observed a constant

Table 1: Running times, RDJ value and error for All-Pair, SampleDJ and TrackDJ for Flickr, and DBLP

(a) Flickr

Data Set Size	**All-Pair**		**SampleDJ**		**TrackDJ**	
n	RDJ	Time (seconds)	Error (%)	Time (seconds)	Error (%)	Time (seconds)
1,000	0.002060	0.08	0.017	34.30 (0.57 min)	0.000	39.33 (0.66 min)
10,000	0.001992	8.82	0.028	40.05 (0.67 min)	0.013	410 (6.84 min)
100,000	0.001992	912.74 (15.21 min)	0.019	90.14 (1.50 min)	0.043	5,253 (1.46 h)
1,000,000	0.001993	97,215.13 (27 h)	0.080	223 (3.72 min)	0.041	51,730 (14.37 h)
Data Set Size	**All-Pair**		**SampleDJ**		**TrackDJ**	
n		Time (seconds)	RDJ	Time (seconds)	RDJ	Time (seconds)
10,000,000	.	*113 days**	0.001998	350 (5.84 min)	0.001997	790,016 (9.14 days)
20,000,000	.	*450 days**	0.002203	246 (4.10 min)	0.002206	1,613,566.20 (18.68 days)
*estimated value						

(b) DBLP

Data Set Size	**All-Pair**		**SampleDJ**		**TrackDJ**	
n	RDJ	Time (seconds)	Error (%)	Time (seconds)	Error (%)	Time (seconds)
1,000	0.005380	0.26	0.053	4.83	0.032	11
10,000	0.005591	7.74	0.307	5.27	0.056	152
100,000	0.005692	580 (10 min)	0.197	7.50	0.078	1,869 (31 min)
1,000,000	0.005653	85,882 (24 h)	0.007	24.64	0.036	29,155 (8 h)
1,256,089	0.005656	135,549 (38 h)	0.061	27.04	0.036	35,879 (10 h)

Table 2: Running times, RDJ value and error for All-Pair, SampleDJ and TrackDJ for synthetic dataset to test effect of data characteristics and error bounds.

(a) Effect of RDJ value

	#ofGroups:GroupSize	**All-Pair**		**SampleDJ**		**TrackDJ**	
n	G:Gs	RDJ	Time(hours)	Error(%)	Time(seconds)	Error(%)	Time(hours)
524,288	32:16384	0.017	5.3	0.34	2	2.05	2.5
524,288	64:8192	0.0087	5.3	0.26	10	1.96	2.5
524,288	128:4096	0.00427	5.3	0.38	39	2.00	2.5
524,288	256:2048	0.00217	5.3	0.02	156	1.95	2.5
524,288	512:1024	0.00105	5.3	0.06	624 (10 min)	1.90	2.5
524,288	1024:512	0.00052	5.3	0.13	2,502 (42 min)	1.91	2.5
524,288	2048:256	0.00026	5.3	0.04	10,089 (3 h)	1.91	2.5
524,288	4096:128	0.00013	5.3	0.39	40,635 (11 h)	2.31	2.5

(b) Effect of error bound ϵ

Error Bound	**SampleDJ**		**TrackDJ**	
ϵ	Time(seconds)	Error(%)	Time(seconds)	Error(%)
0.2	0.3	0.098	5	0.022
0.1	1	0.021	20	0.369
0.05	4	0.015	80	0.003
0.025	15	0.016	318	0.004
0.0125	58	0.003	1271	0.081

running time (approx. 5 min for the Flickr dataset), consistent with our theoretical findings in Section 4.3. Also consistent with our theoretical findings, TrackDJ shows linear behavior and outperforms All-Pair for datasets of size n larger than 1 million. Note that for the approximation algorithms TrackDJ and SampleDJ the actual error is clearly below the defined error bound (in most cases less than 0.001 for ϵ = 0.1) For the given datasets, SampleDJ shows much better performance than TrackDJ; however, in the next paragraph we will see that TrackDJ can be more efficient for data distributions with very high diversity.

5.3.3 Effect of Data Characteristics and Error Bound

We have seen that SampleDJ always provides accurate estimates within short running time. This is because the RDJ value is still relatively large. However, Table 2a shows SampleDJ degrades indeed in $O(\frac{1}{RDJ^2})$ rate for varying RDJ values on our synthetic data as shown in our theoretical analysis. In comparison, All-Pair and TrackDJ are not sensitive to the RDJ values; TrackDJ outperforms SampleDJ if RDJ values are small.

In another set of experiments, we used the first 1000 lines of the DBLP data to test the effect of the relative error bound ϵ on running time and accuracies of SampleDJ and TrackDJ. We set δ to a fixed vale of 0.001 and varied ϵ. Table 2b confirms that the running time of both SampleDJ and TrackDJ are indeed $O(\frac{1}{\epsilon^2})$. Ideally, the errors of both solutions should strictly decrease when the error bound decreases, but in fact this is not the case in our experiments and the error of both solutions fluctuate sometimes. We believe this is because the real error is already quite small and close to the optimal. Note that even the best possible pseudo random number generator and min-wise hashing implementation are approximations to the theoretical ideals.

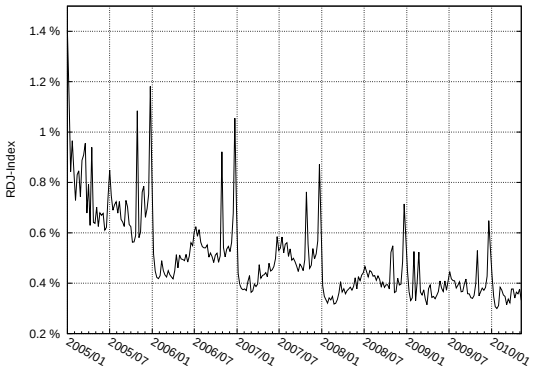


Figure 1: Flickr photo tags similarity over the time period 2005-2010, revealing a trend towards higher annotation diversity throughout the years, as well as periodic peaks due to recurring annual events.

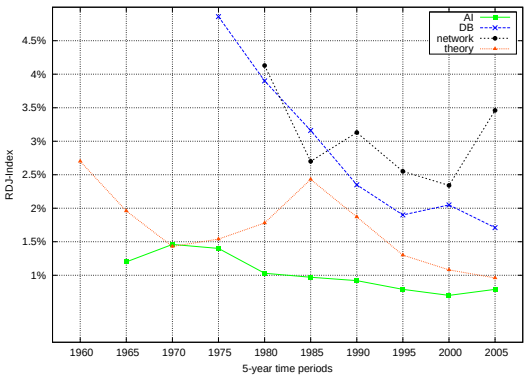


Figure 2: Topic diversity vs. 5-year interval

5.4 Summary

The main characteristics of the tested algorithms can be summarized as follows:

- *All-Pair* (Straightforward): Running time is predictable but can be too slow for large data sets; directly applicable for other similarity measures; running time increases in a quadratic rate with the number of objects in the data set; can be still viable in practice for smaller datasets.
- *SampleDJ*: Extremely fast for the real data sets we used regardless of the data set size; directly applicable for other similarity measures; can be too slow in some cases depending on the diversity value; running time is not predictable without prior knowledge about the input. A practical solution might be to try SampleDJ first and use TrackDJ in case of failure.
- *TrackDJ*: Predictable $O(n)$ running time regardless of the input; applicable for similarity measures for which there exist hash functions with the LSH property (requiring additional analysis with respect to performance guarantees); solves the $O\left(n^2\right)$ running time problem which makes the diversity index practically computable even for large data sets.

Note, that all three approaches can be parallelized (cf. Section 4).

6. CHARACTERIZING THE DIVERSITY IN CORPORA: EXAMPLE STUDIES

We now show the results of sample studies on a number of real world datasets. *Note, that low RDJ values correspond to high diversity.*

6.1 Development of Flickr over Time

We also studied the temporal development of tag annotation diversity in the social photo sharing site Flickr over the time period from 01 Jan 2005 until 05 Sept 2010 (see Section 5.1 for a description of the Flickr sample).

Our first observation from Figure 1 is a year wise increase in annotation diversity reflected by the decrease of the RDJ values. A possible explanation for the increasing topic diversity is that the user community has broadened over the years with Flickr becoming more and more popular for people of different ages, origins, and backgrounds.

Secondly, exploring diversity on a more fine-grained, monthly level reveals additional interesting patterns and periodicities. In order to obtain representative tags for the most interesting parts of the curve, we computed Mutual Information (MI) [19] for each tag, which measures how much a joint distribution of terms and categories deviates from a hypothetical distribution in which terms and categories (time period for a peak and the rest of the year in our case) are independent of each other. Note that the number of photos captured for this figure remains rather constant over time, amounting to approx. 10,000 images per day. The RDJ value curve starts at its minimum at the beginning of each year (January) and remains at the lowest level till the end of March, reflecting high topic diversity. Photos in this time period are mainly described by tags for a rather broad range of topics like *winter*, *snow*, *vacation*, or *house*. In summer, the curve starts to increase steadily, and reaches its maximum in mid-July before it decreases steadily until the end of September, with the most representative tags in this time period being *graduation*, *wedding* and *beach*. In October the RDJ value increases sharply, reaching a peak by the end of that month. Our term analysis reveals that this is due to the popular holidays *halloween* and *thanksgiving*. Finally, the RDJ curve reaches its maximum at the end of December where Christmas is the dominating topic.

6.2 DBLP Bibliography

To study diversity in the DBLP dataset, we classified Computer Science (CS) paper titles into groups based on publication year and publication venue. We intended to explore how topic diversities of Computer Science papers change with time and research area. We picked well-known publication venues to represent 4 research areas: Databases (DB), Artificial Intelligence (AI), Computing Theory and Computer Networks. The venues representing the areas are as follows: DB: *SIGMOD*, *VLDB*, AI: *AAAI*, *IJCAI*, Theory: *STOC*, *FOCS*, and Computer Networking: *SIGCOMM*, *INFOCOM*. Figure 2 shows how topic diversity changes with time. The figures indicate that the topic diversities of CS papers increased gradually over time with the exception of Network papers within the past 5 years. Also, curiosity driven papers (Theory, AI) seem to have higher topic diversities than application driven ones (DB, Networking).

6.3 Diversity in Clustered Data

Cluster analysis or "clustering" refers to the division of a set of objects into subsets (called clusters) so that objects in the same cluster are more similar and objects in different clusters are less similar. In many contexts unsupervised machine learning techniques like hierarchical, partitional or spectral clustering are employed to achieve this goal. Intuitively, the higher the number of clusters in a particular data set, the higher its diversity. In this section we want to verify if this property is reflected by the RDJ index. To this end, we analyzed three real world data sets:

Table 3: Size, title and RDJ value (in percent) per category in Reuters Corpus, per group in Flickr-Groups collection, and per cluster in US-Census data

(a) Reuters RCV1 Categories

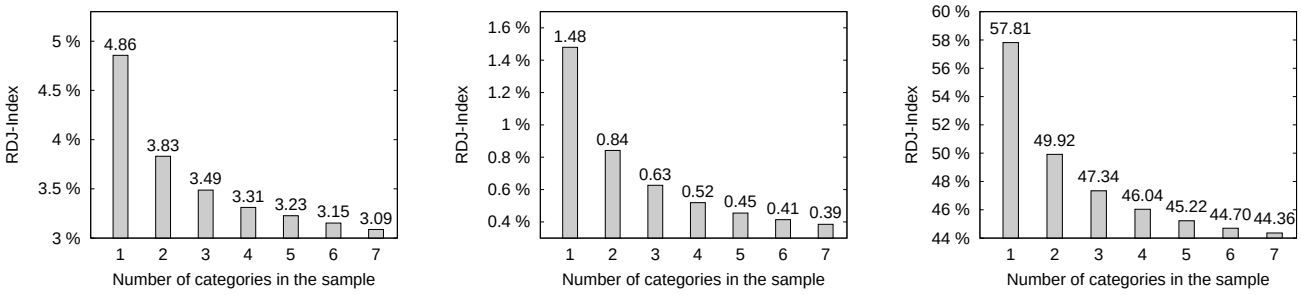
Size	News Category	RDJ
299,612	Corporate/Industrial	4.31
204,820	Markets	4.79
66,339	Economics	4.69
35,769	Government/Social	5.73
35,279	Sports	3.45
33,969	Domestic Politics	5.21
31,328	War, Civil War	5.81

(b) Flickr Groups

Size	Group Title	RDJ
139,344	Pictures Of England	1.63
121,391	Dark Art	0.57
98,901	Aircraft Photos	1.99
89,606	Absolutely beautiful	0.51
76,265	Visual Arts!!	0.61
73,632	Lonely Planet:'Leaving'	0.48
71,158	Lighthouse Lovers	4.56

(c) UCI US-Census Education Based Clusters

Size	Educational Background	RDJ
562,837	High School, Diploma or Ged	49.41
366,116	Some College, But No Degree	49.22
273,281	5th, 6th, 7th, or 8th Grade	51.63
213,941	Bachelors Degree	51.36
174,653	1st, 2nd, 3rd, or 4th Grade	70.97
109,834	N/a Less Than 3 Yrs. Old	84.55
107,142	10th Grade	47.56



(a) Reuters RCV1 (b) Flickr Groups (c) UCI US-Census

Figure 3: Diversity increases with a growing number of categories in Reuters Corpus, groups in Flickr-Groups, and educational levels in US-Census datasets

1. *RCV1:* The "Reuters Corpus Volume 1" (RCV1) is a corpus of news feeds released for public research in 2000 [16], and employed in several works on clustering and classification of text documents [26, 19, 23]. RCV1 consists of text documents divided into news categories like "sports", "politics" and "economics" which we used as cluster ID's.
2. *Flickr Groups:* The second dataset consists of tagged photos from different interest groups in Flickr like "Pictures of England" or "Aircraft Photos" and general groups like "Absolutely Beautiful" or 'Visual Arts". Here we used the group IDs as cluster IDs and image tags as textual content.
3. *US Census 1990:* In demographics clustering is defined as the gathering of various populations based on factors such as ethnicity, economics, or religion. For our experiments we used the US Census Data from 1990 [20] consisting of details from around 2.5 millions US citizens[5]. Attributes include, for instance, *income*, *dedication* and *educational background*. We selected the latter for obtaining the clusters, with examples of cluster IDs being "High School Graduate", "Diploma", or "Bachelor Degree".

Table 3 provides an overview over the topics in the datasets and the number of instances per topic. For each dataset we selected the top $n = 7$ largest clusters. We computed sets of all $\binom{n}{k}$ possible cluster combinations for $k = 1, \ldots, n$ clusters. In order to obtain balanced cluster sizes, we restricted the number of instances, s, per combination to the size of the smallest cluster, and randomly selected $\frac{s}{k}$ instances from each cluster in the sample. For the RCV1, Flickr and UCI Census datasets we obtained 31,328, 71,158 and 107,142 instances, respectively. Finally, for each number of clusters k we computed the average RDJ index value across all cluster combinations.

Figure 3 shows the RDJ index vs. the number k of clusters contained in the sample sets. *The key observation is that the RDJ value indeed decreases with the number of clusters.* This shows that topic diversity in the sample sets is mirrored by the Jaccard-based RDJ index. Despite of the large structural and conceptual differences between the distinct corpora and the large differences between the absolute RDJ values, the decreasing pattern observed is remarkably similar across corpora.

A comparison of diversity values in different corpora and for specific clusters reveals further interesting insights (cf. Table 3). Generally, the diversity in the Flickr data set is highest (corresponding to lower RDJ values) as the tags used for diversity computation can be defined by users and are not restricted. The Reuters data set contains more restricted vocabulary, and is less diverse. Finally, attributes in the UCI Dataset are well defined, the number of possible terms per instance is small (68), and the "vocabulary" limited, which results in high RDJ values. We also studied the correspondence of RDJ diversity values to different topics. For example in Table 3a the first five categories including "Industrial" (*RDJ=4.31*) or "Markets" (*4.79*) are more general and therefore more diverse (an exception being "Government/Social" with (*RDJ=5.73*)). "Domestic Politics" (*5.21*) and "War, Civil War" (*5.81*) are sub-categories that are more specific and less diverse. The Flickr groups "Visual Arts" (*0.61*) and "Absolutely beautiful" (*0.51*) are more general than "Lighthouse Lovers" (*4.56*) and "Aircraft Photos" (*1.99*). Finally, for the census data, we observe that the group "N/a Less Than 3 Yrs. Old" (*84.55*), which mainly represents babies, has the lowest diversity.

The RDJ index clearly corresponds to the number of clusters in a data set and could be used in efficient heuristics for determining the number of clusters in large data sets which is required as a parameter in many clustering techniques such as k-means [1], k-medoids or the expectation-maximization algorithm for clustering. In our future work we plan to investigate in depth how exactly different diversity indexes might help to estimate parameters for clustering in large datasets.

7. SUMMARY

Contributions: Two novel, efficient algorithms for estimating the diversity of whole corpora with probabilistic guarantees, and illustrative example studies on different datasets. To the best of our knowledge, there exist no other methods for efficiently computing the diversity of whole corpora.

[5]The set is available in the UCI machine learning repository (`http://archive.ics.uci.edu/ml/index.html`).

Simplicity of the diversity measure: We would like to point out that nearly all recent diversity measures used in the context of query result diversification and cited in the paper are actually based on "simple" pairwise similarity or distance computations. We consider "simplicity" as an asset since simple ideas have a higher chance to be applied in practice. For instance, one of the reasons why relational databases are so popular is that the relational model is simple.

TrackDJ vs. SampleDJ: Although our SampleDJ algorithm outperforms TrackDJ for our specific datasets, it becomes superior for datasets with smaller RDJ value. In this paper we have presented a strong theoretical result, which has the potential to be applied in additional contexts: "Average pair-wise similarity can be computed in one pass with probabilistic guarantees."

Tuning parameters: The parameters of the algorithms are basically the user-defined epsilon and delta which we comprehensively study in our experiments. As described in Section 4, window size W for SampleDJ is independent of data set sizes; the number of windows is determined *automatically*, and changes according to the data properties. Therefore, no tuning of W is required.

Vocabulary size: Guarantees for error bounds and running times for both SampleDJ and TrackDJ do not require any assumptions about the overall vocabulary size of the corpus.

Parallelization: We elaborate on parallelization for the three considered algorithms (cf. Section 4 with an additional backward reference made in the summary of Section 5). All of the algorithms can run approx. k times faster on k machines.

Main memory consumption: TrackDJ allows for computing diversity in one pass, thus each term-set needs to be read just once, making the algorithm feasible even in case the main memory size is limited. SampleDJ requires only a fraction of the comparisons of the All-Pairs algorithm, and thus allocates only a negligible amount of main memory. In case hard-drive access is necessary, SampleDJ would further save a considerable number of I/O processes compared to the baseline approach.

Future work: We aim at adapting our algorithms to other similarity measures like Euclidean distance and cosine similarity by using LSH and its variants. In addition, for other multimedia types such as videos or photos a study of diversity based on visual properties (e.g. color distributions or shapes) may reveal further insights about social content sharing environments such as Flickr or YouTube.

8. ACKNOWLEDGMENTS

This work is partly funded by the European Commission under the grant agreement 270239 (ARCOMEM) and 287704 (CUBRIK) as well as by the NTH School for IT Ecosystems. We would like to thank Dr. Davood Rafiei for helpful and insightful discussions at the initial stage of the work.

9. REFERENCES

[1] Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. PODS '02, Madison, Wisconsin.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, New York, 1999.

[4] A. Z. Broder. Min-wise independent permutations: Theory and practice. ICALP '00, London, UK.

[5] A. Z. Broder. On the resemblance and containment of documents. SEQUENCES '97, Washington, USA.

[6] A. Z. Broder. Identifying and filtering near-duplicate documents. COM '00, London, UK, 2000.

[7] A. Z. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *J. Comput. Syst. Sci.*, 60:630–659, June 2000.

[8] M. Charikar. Similarity estimation techniques from rounding algorithms. STOC '02.

[9] P. Dagum, R. Karp, M. Luby, and S. Ross. An optimal algorithm for monte carlo estimation. *SIAM J. Comput.*, 29:1484–1496, March 2000.

[10] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. SCG '04, New York, USA.

[11] J. D. Fearon. Ethnic and cultural diversity by country*. *Journal of Economic Growth*, 8:195–222, 2003.

[12] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. WWW '09, Madrid, Spain.

[13] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. STOC '98, Dallas, Texas, USA.

[14] C. C. Krebs. *Ecological Methodology*. HarperCollins, 1989.

[15] C. Lévêque and J.-C. Mounolou. *Biodiversity*. John Wiley & Sons, 2003.

[16] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[17] M. Ley. The dblp computer science bibliography. `http://www.informatik.uni-trier.de/~ley/db/`.

[18] S. Lieberson. Measuring population diversity. *American Sociological Review*, 34(6):850–862, 1969.

[19] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[20] C. Meek, B. Thiesson, and D. Heckerman. The learning-curve sampling method applied to model-based clustering. *J. Mach. Learn. Res.*, 2:397–418, March 2002.

[21] E. Minack, W. Siberski, and W. Nejdl. Incremental diversification for very large sets: a streaming-based approach. In *SIGIR '11*, Beijing, China.

[22] Olken. Random sampling from databases. In *Ph.D. Diss. (University of California at Berkeley)*, 1993.

[23] O. Papapetrou, W. Siberski, and N. Fuhr. Text clustering for peer-to-peer networks with probabilistic guarantees. LNCS, pages V.5993, 293–305. Springer Berlin / Heidelberg, 2010.

[24] D. Rafiei, K. Bharat, and A. Shukla. Diversifying web search results. In *WWW '10*, Raleigh, USA.

[25] I. Rafols and M. Meyer. Diversity and network coherence as indicators of interdisciplinarity: case studies in bionanoscience. *Scientometrics*, 82(2):263–287, 2010.

[26] N. Sahoo, J. Callan, R. Krishnan, G. Duncan, and R. Padman. Incremental hierarchical clustering of text documents. In *CIKM '06*.

[27] E. H. Simpson. Measurement of diversity. *Nature*, 163, 1949.

[28] A. Stirling. A general framework for analysing diversity in science, technology and society. *Journal of The Royal Society Interface*, 4(15):707–719, 2007.

[29] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. A. Yahia. Efficient computation of diverse query results. In *ICDE'08*, Washington, DC, USA.

[30] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW '05*, New York, USA.