Dimensionality Reduction for Documents with Nearest Neighbor Queries

Stephen Ingram and Tamara Munzner

Abstract

Document collections are often stored as sets of sparse, high-dimensional feature vectors. Performing dimensionality reduction (DR) on such high-dimensional datasets for the purposes of visualization presents algorithmic and qualitative challenges for existing DR techniques. We propose the Q-SNE algorithm for dimensionality reduction of document data, combining the scalable probability-based layout approach of BH-SNE with an improved component to calculate approximate nearest neighbors, using the query-based APQ approach that exploits an impact-ordered inverted file. We provide thorough experimental evidence that Q-SNE yields substantial quality improvements for layouts of large document collections with commensurate speed. Our experiments were conducted with six real-world benchmark datasets that range up to millions of documents and terms, and compare against three alternatives for nearest neighbor search and five alternatives for dimensionality reduction.

Keywords: dimensionality reduction, document visualization

1. Introduction

Dimensionality Reduction (DR) algorithms transform a set of N high-dimensional input data points with M dimensions into an output dataset with the same number of data points but a *reduced* number of m dimensions: $m < M$. We focus here on the use of DR for visual data analysis, in contrast to other possible goals such as generating tractable datasets for further algorithmic processing. We further focus on the common choice where m is equal to 2 and the low-dimensional data is shown as a scatterplot, following guidelines for visual encoding [1, 2].

A frequent use case for visual data analysis with DR is for the exploration of large document collections. These collections range in size from thousands to millions of documents, precluding the simple approach of simply expecting the analyst to read through each of them. Each of the N points represents a document, whose contents are represented as a vector of M terms. These datasets are very high dimensional, where there are typically at least as many terms as documents; in these cases, where $N = M$, the term vectors are sparse.

Despite the popularity of using DR for documents, we have identified a major drawback of existing DR algorithms that leads to low-quality results for document collections because of the sparse nature of these datasets. We present a new DR algorithm, Querying Stochastic Neighborhood Embedding (Q-SNE), that yields significantly higher-quality results for large document collections than previous work, at commensurate speeds. Our design target is a system that scales to millions of documents and dimensions with tractable speed and memory use. The main innovation of Q-SNE is to compute approximate nearest neighbors using the APQ [3] impact-ordered inverted-file approach inspired by query-oriented techniques from the information retrieval literature, in contrast to the previous BH-SNE [4] approach that finds nearest neighbors using a combination of PCA and Vantage-Point trees [5].

We validate our claims thoroughly with computational experiments on six benchmark document collection datasets that range up to two million documents and up to one and a half million terms. Though there are many possible definitions of layout quality, we follow the reasoning of Venna et al [6] and focus on neighborhood preservation, using precision/recall curves as a quantitative metric for layout quality. First, we compare the nearest-neighbor computation to three competing approaches in terms of both quality and speed for this document data, and show dramatic improvements in both. Second, we compare the quality and speed of the Q-SNE layout to five other DR algorithms: PCA [7], Glimmer [8], LSP [9], LAMP [10], and BH-SNE [4]. We show that Q-SNE has equivalent speed to and higher quality than BH-SNE, and both of them outperform the others. Finally, we illuminate the qualitative differences between the Q-SNE and BH-SNE layouts with a detailed analysis of how and why Q-SNE more accurately reflects the neighborhood structure of the data.

The contribution of our work is both the Q-SNE algorithm itself, and the extensive experimentation comparing the performance of the algorithm and its components to previous work.

In Section 2 we discuss the challenges of sparseness in document collections, and characterize how four classes of previously proposed DR algorithms handle its implications. In Section 3 we discuss multiple interpretations of the concepts of queries and neighborhoods, and characterize how three major approaches to nearest neighbor search handle large document collections. We then present the

Q-SNE algorithm itself in 4. Section 5 presents extensive experimental evaluation of our performance claims, and we conclude in Section 6.

2. Document Data and Previous DR Algorithms

The usual strategy for transforming documents into high-dimensional data is to map each document to a vector in *term space*, the vector space where each dimension represents a weight assigned by a term-scoring method. The classic TF-IDF technique [11] assigns each of a document's term dimensions a weight proportional to the frequency of the term in the document. This term weight is then discounted by the frequency of the term across all the documents. There are many possible vector similarity metrics in use. A popular one in information retrieval is the cosine similarity: the inner product of the two vectors, normalized by their lengths to range between 0 and 1. Most documents in a large collection have little to no similarity to each other because they have little or no overlap in the terms that they contain. Thus, most pairwise similarities are exactly 0 or very close to 0, and the term vectors are considered to be sparse data and handled accordingly.

Many DR algorithms require distances, rather than similarities, where cosine distance is simply 1 minus cosine similarity. Thus, the preponderance of pairwise distances are exactly 1 or close to 1 in large document collections. The semantics of these numerous unit-length distances is important because conceptually they correspond to infinite distances. However, many previous DR algorithms fail to handle these infinite distances appropriately, resulting in impaired quality because of layout distortion, or wasted time because of useless computations.

We classify previous DR algorithms into four major classes based on the approach to optimization: orthogonal projections, global distances, manifold distances, and probability distributions. We now discuss in detail how each of these families of algorithms handles these unit-length distances; only the probability distributions family has a mathematical framework that does so gracefully.

2.1. DR with Orthogonal Projections

The family of orthogonal projections includes methods such as principal component analysis (PCA) [7] which computes the linear projection of coordinate data onto an orthonormal basis that best preserves the variance of the data. The transformation has numerous useful properties, both by being the linear projection of the data into the low-dimensional space with least-squared error, and also by providing a method for back-projection to the original embedding space of the data.

Variants of PCA algorithms exist to address speed concerns and be computed very efficiently using approximation algorithms of the singular value decomposition of a matrix [12].

Orthogonal projections are often an ineffective solution for visualizing document data because the majority of the variance in document data matrices is captured by the unit length distances arising from the orthogonal relationships between term vectors in the data. Assuming the data is clustered within G orthogonal subspaces, the average distance between these clusters will be 1.0, creating a G-dimensional simplex. The first G principal components will primarily express these unit-length, inter-cluster distance relationships, often without resolving any of the intra-cluster distance relationships that may reside in a subset of orthogonal dimensions. Because G is often larger than 2 or 3, PCA by itself is an ineffective general tool for producing low-dimensional visualizations of document collections.

2.2. DR with Global Distances

Global distance methods, such as Multidimensional Scaling (MDS) [13], compute a low-dimensional layout of points with inter-point distances that best match the input high-dimensional distance matrix. There is surprising variability to the number of MDS techniques [14], but our arguments here hold for any particular MDS algorithm. The problem that MDS methods have with document data lies with optimizing the layout to fit the preponderance of unit length distances that arise when two documents share no terms. MDS methods optimize a function called *stress*, which sums the residuals between the high and low dimensional distance matrices.

Buja and Swayne noted the problem of *indifferentiation* [15] when using MDS for datasets where distances are clustered around a positive constant, with the extreme case being a perfect simplex: the high-dimensional analog of a tetrahedron, where every object is equally distant from every other object. In this case, applying MDS yields low-dimensional layouts with a characteristic shape approximating a uniform ball: in 2D, it is a circular disk with a sharp edge and a low density in the center. They caution data analysts to avoid misinterpreting this artifact as meaningful dataset structure.

Continuing with this line of analysis, we argue that MDS algorithms spend most of their time on nearly useless computations when applied to document data. Because no simplex can fit without distortion into a low-dimensional projection, the unit-length distances make up the majority of the error of the objective function. Unfortunately, these disconnected unit-length distances dominate the computation in a way that is inversely related to their importance; the important local distances to their nearest neighbors are not given enough weight.

One strategy to improve MDS maps is to modifying the objective function using polynomial re-weighting of the terms inversely proportional to distance, as in Sammon mapping [16]. However, such schemes do not go far enough; they still retain the need to fit all the distances of unit length in the objective function, as shown by Paulovich [9].

2.3. DR with Manifold Methods

Manifold distance dimensionality reduction methods preserve distances across local surfaces formed by the data in high dimensional space, building a model of manifold connectivity. The early work such as the Isomap algorithm [17], Local Linear Embedding [18], and Laplacian Eigenmaps [19] has been followed by many other variants [20, 21, 22, 23].

Manifold methods are computationally intensive and often subject to numerous tuning parameters controlling how the local manifold is inferred from the data. Furthermore, several methods make assumptions about the sampling density and number of manifolds generating the data points under analysis. Most of these methods are targeted at the ideal case of data with smooth, uniformly sampled, nonlinear structures [24].

While the notion of stitching together the term vectors into a useful layout may seem attractive, in practice manifold techniques struggle with building an effective manifold model over vectors derived from document collections. Their connectivity assumptions often do not hold because the sampling of the different subspaces is very sparse, often leading to significant distortions in the manifold distance calculations. This phenomenon is well documented by van der Maaten [25], who reports that the performance of manifold methods are consistently thwarted by noisy, real-world examples.

2.4. DR with Probability-based Methods

Probability-based dimensionality reduction methods such as SNE [26], t-SNE [27], NERV [6], NE [28] and BH-SNE [4] try to minimize the discrepancy between high dimensional and low dimensional probabilities derived from distances. This discrepancy is measured as the Kullback-Leibler divergence

$$\sum_{i}^{N}\sum_{j}^{N} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{1}$$

between the two distributions P and Q representing the distributions for the high-dimensional and low-dimensional cases respectively. Probability methods work by building conditional probability distributions for each point over the other data points based on high-dimensional distances, where probability is interpreted as the likelihood that point i will be chosen as a given point j's nearest neighbor. This approach intuitively assigns points with close distances a high probability value, and points with far distances a low probability value.

Probability-based methods have objective functions that exhibit strong similarity to the Stress function, used to optimize global distance, but we argue they are different enough to warrant special categorization. The main distinction between MDS and probabilistic methods is best illustrated by comparing their objective functions. Stress, and its descendants, are expressed as a weighted sum of squared residuals between high and low distances

$$\sum_{i<j<n} w_{ij}(d_{ij} - \delta_{ij})^2$$

where the weights w_{ij} and δ_{ij} are *fixed* inputs into the objective function. In contrast, the individual layout terms of the SNE objective function in Equation 1, the $q_{j|i}$'s, are functions of the locations of all the other points in the layout, not just point j and i. Therefore, even ignoring the log transformation, no selection of weights w_{ij} or function of the input distances δ_{ij} can always reproduce the results of SNE or other probabilistic dimensionality reduction methods.

The t-SNE probability-based method produces some of the most visually salient cluster visualizations of high-dimensional data. It accomplishes this salience by mapping the high-dimensional Gaussian probability distribution to a heavier-tailed Student-t probability distribution in the low-dimensional space. The mismatch in tails is specifically designed to address the *crowding problem* [27], where mappings of points within a sphere of radius r high-dimensional space quickly exhaust the exponentially smaller volume contained within a corresponding sphere with identical radius in low-dimensional space.

Unlike other DR techniques, Probability-based methods can implicitly handle the unit-length distances induced by having zero cosine similarity. This is done by mapping unit-length distances to zero probabilities. Unlike the previously described DR techniques, probability methods permit significant flexibility in the relative placement and handling of zero-probability points. Using a spring-force metaphor, zero-probability points are attached to a point with an infinite-length spring that rapidly diminishes in spring force as a function of distance. In contrast, distance-based methods such as MDS always assign a fixed-length spring between points that will invariably contract to some finite length, producing an attractive force between two points that have no meaningful relationship.

2.5. Document Oriented DR

Numerous DR techniques specialize in computing layouts of high-dimensional document vectors. The IDMAP algorithm of Minghim et al [29] performs dimensionality reduction using the FastMap linear projection technique [30] followed by a force-based relaxation step. The LSP algorithm of Paulovich et al [9] first selects a set of control points with k-medoids [31] and then places the remaining points with a linear mapping that best preserves nearest-neighborhood membership. Paulovich and Minghim's HiPP algorithm[32] uses a hierarchical clustering of the data to recursively project the data with the LSP technique into disjoint circular regions. Both LSP and HiPP were validated by using a metric called "Neighborhood Preservation", which is equivalent to precision-recall when the sizes of relevant and retrieved documents sets are identical. In contrast to these existing techniques, our approach avoids any linear projections or distance-based positioning, requires no selection of control points, and imposes no hierarchical structure of the data.

3. Queries and Previous Nearest Neighbor Algorithms

Venna et al [6] suggest framing the use of DR for visualization, where people examine the visible structure of the low-dimensional projection as a 2D scatterplot, using terms from the information retrieval (IR) literature. In IR, the goal is optimize the true information content retrieved by a *query*. In their framing, Venna et al define a query to be a point selected by a data analyst, and the result retrieved by the query to be the set of neighboring points around the query point. They point out that analysis of the *neighborhood* in the low-dimensional projection compared to the high-dimensional space in terms of true positives, false positives, and misses matches up with common quality measures of *precision* and *recall* [33] in IR, that compare the number of retrieved points to the number of relevant points. As a systematic way to compare the visual neighborhood set in the projection to the true neighborhood set in the high-dimensional space, they propose quantitative measurement of DR layout quality using precision-recall (PR) curves, where the number of retrieved points is varied to create the curve. We adopt their proposed methodology in this paper, using PR curves for quantitative comparative analysis of DR layouts in Section 5.

We also consider the concept of a *query* and a *neighborhood* in a different context: in the literature on nearest neighbor (NN) search, the goal is framed as finding the neighbors for a query point. We use approximate NN search as a component of the Q-SNE DR algorithm. In particular, we use approximate K-nearest neighbor search compute the set of k nearest points in a data set for a set of query points. Our design target is scalability to queries for millions of documents in under an hour on a single CPU. We chose APQ [3] as a suitable algorithm, as discussed below. APQ uses internal data structures that efficiently support generating all pairs of queries; with APQ, the query set is equal to the entire data set, not just a single point as above.

We now discuss in more detail three approaches to NN search: spatial partitioning, mapping, and queries. We provide the rationale behind our choice of the query-based method APQ [3].

3.1. NN with Spatial Partitioning

Spatial partitioning strategies conduct NN search by constructing hierarchical structures that partition the data space. Examples include balance-box decomposition trees [34], k-d trees [35], cover trees [36], and vantage point trees [5]. After constructing hierarchical spatial decompositions of the data space with such methods, a spatial search for nearby points becomes similar to a generic search tree traversal.

This spatial approach has two shortcomings with respect to the very high-dimensional spaces that occur with document term vectors. First, in axis-aligned trees such as k-d trees, the sheer number of dimensions in the data results in trees that are too large to accelerate nearest neighbor search. Second, metric trees like vantage point trees and cover trees are confused by the cosine distance because it is not a true distance metric as it does not obey the triangle inequality. The BH-SNE DR algorithm does use vantage point trees, but only after a pre-processing step that reduces the number of dimensions to 50 with PCA. We conjecture that this approach discards potentially important information; our goal with Q-SNE is to use a scalable one-step approach that exploits all of the useful information while avoiding unnecessary computation. We deem that the spatial partitioning NN approaches are not suitable for this goal.

3.2. NN with Mapping

Mapping-based techniques, by contrast, build functions that map data points close together with high probability. Using this technique, the search is restricted to the set of bins to which similar points are mapped. The seminal example of mapping techniques is Locality Sensitive Hashing (LSH) [37], which uses random projections to divide the input space into a set of bins. Mapping techniques share a limitation with spatial partitioning techniques: they operate without prior knowledge of the subspace structure of the data. In contrast, the query-based approaches described below build data structures that exploit useful structure directly, rather than inferring it through time-consuming computation. These approaches outperform mapping techniques for sparse data such as document collections.

3.3. NN with Queries

The problem of exact NN search has been tackled in the IR and database literatures using a data structure called the inverted file [38] that maps from a single query term to the list of documents that store it, in an inversion from the standard approach of storing a term vector that maps from a single document to the list of all the terms it contains. Inverted files accelerate query processing by providing easy access to only those documents that share the query term, avoiding the need to check against the entire collection. They provide benefits only when the data is sparse, as in document collections; with dense datasets, the inversion would simply add additional overhead rather than speed computation.

Initial work in IR reduced the search space of nearest points by partially ordering the inverted file, and then iteratively computing upper bounds on the remaining documents to be processed [39, 40]. Recent work focused on scaling up similarity calculations to massive datasets also uses a bound calculation to compute the exact nearest neighbors, while also building the inverted file index on the fly [41][42].

An efficient approach to approximate NN search is by reordering and traversing these lists according to impact, so that high-scoring queries are found earlier in the search process than low-scoring ones [43]. The advantage of impact-ordered traversal is that the computation can be truncated after the desired number of neighbors is found; full traversal is not required.

We recently proposed the All Pairs Query (APQ) algorithm [3] for impact-ordered, inverted-file approximate k-NN search. APQ constructs the approximate set of k nearest neighbors by performing an ensemble of queries across all pairs of data points, as the name suggests. (In the database literature, the all-pairs k-nearest-neighbor problem is referred to as a top-k similarity join [44].) APQ is explicitly designed to exploit the sparse structure of document collections to avoid the need to exhaustively compute all of the similarities between data points. Each term vector resides in a low-dimensional subspace of term space, and the inverted file maintains a record of the subspace dimensions for each vector. In APQ, the traversal of the documents indexed by the inverted-file terms contained in the query is ordered by the magnitude of their 1D projections with the query's subspace dimensions. Thus, when using the terms of a given term vector as a query into the impact-ordered index, we first encounter those vectors whose projections onto the query's subspace dimensions are the largest. These vectors are more likely to be in the set of points closest to our query. Thus the impact-sorted, inverted file provides the equivalent of a mapping function, but without the need to derive partitions or hash functions from the data through computation. The contribution of our work in this paper is to incorporate APQ into a DR algorithm.

4. Q-SNE Algorithm

Our design target for Q-SNE was scalability for datasets with millions of documents and millions of terms, achieving significantly higher quality layouts than previous work for document data while preserving commensurate computation times.

The Q-SNE algorithm has three phases: document processing, NN calculation, and layout. Q-SNE builds on Barnes-Hut Stochastic Neighborhood Embedding (BH-SNE) [4], which itself builds on Stochastic Neighborhood Embedding (SNE) [26]. The major difference between Q-SNE and BH-SNE is in the NN calculation phase, where we use the APQ algorithm rather than the combination of PCA and Vantage Point Trees.

In the first phase, documents are converted into term vectors using the TF-IDF vector scoring function. In the second phase, the set of 30 approximate nearest neighbors and the distances to them are computed for each document using the APQ algorithm [3]. In the third phase, these nearest neighbors are used to produce a layout by minimizing the t-SNE objective function [27], using the same momentum-based gradient descent with Barnes-Hut quadtree approximation as used in the BH-SNE algorithm [4] and terminating after a fixed number of iterations.

We now discuss these three phases in more detail.

4.1. Document Processing

To efficiently calculate similarities between documents, we convert the documents into vectors in term space, where each dimension represents a term. To calculate the weights of each dimension we select the classic TF-IDF scoring function [11]. The resulting dimensionality of term space is often very large; we show results for benchmark datasets with term counts ranging from thousands to millions in Section 5.

The major scalability challenge in this phase is memory usage. While previous work such as BH-SNE does not include document processing as part of the algorithmic pipeline, we consider the processing stage worthy of a brief discussion given the significant engineering requirements of handling very large collections of document data.

It is crucial to efficiently manage document vector storage; many straightforward approaches to processing documents do not scale to collections with millions of terms. The Q-SNE processing stage converts directories of text documents into sparse TF-IDF feature vectors stored in a Compressed Sparse Row matrix format[45]. We build on the sparse matrix storage support of the scikit-learn toolkit [46], which also provides a standard set of English stop words and generates bigrams.

4.2. Nearest Neighbor Calculation

The approximate nearest neighbor approach used in BH-SNE is to first reduce the dataset to 50 dimensions using PCA and then use the Vantage Point metric tree (VPTree) algorithm for approximate nearest neighbor search. We claim that this approach leads to inaccuracy for sufficiently large document collections, and show experimental evidence to validate this claim in Section 5.2.

The major scalability challenges in this phase are both computational speed and result quality. The key insight in this paper is that an alternative approach to approximate NN search could yield much better results for DR when the data is a set of very sparse, high-dimensional document vectors. As we discussed in Section 3, the limitations that spatial partitioning and mapping approaches to NN must infer the subspace structure of the data are addressed by using a query-based approach that features an impact-ordered inverted file as the core data structure. We thus select the APQ algorithm [3], which exploits the sparse structure of document datasets to compute a high-quality approximation of the k nearest neighbors quickly. Our hypothesis is that higher quality NN results will lead to a higher quality layout for Q-SNE than BH-SNE, despite using essentially the same approach for the layout computation itself. We show experimental evidence to validate this hypothesis in Section 5.3.

The only parameter required for APQ is the number of nearest neighbors to compute. The parameter setting of $k = 30$ was an empirical choice, based on our observation that it produced stable results across the entire range of benchmark datasets described in Section 5.1.

4.3. Layout

We select the probabilistic t-SNE objective function for computing our document embedding [27]. The choice of a probabilistic method gracefully transforms the large number of unit length distances in document collections into zero probabilities, as discussed in Section 2.4. In addition, t-SNE's use of mismatched tails for cluster separations creates visual salience for the disparate clusters, and cluster-based analysis is a typical approach used in exploring large document collections.

The major scalability challenge in this phase is speed, requiring to another choice of approximation methods during the layout computation itself. Probabilistic DR methods optimize the KL Divergence between the input P and visual Q distributions of data points. In practice, this optimization is done with either momentum-weighted gradient descent[27] or conjugate gradients[6], both of which require calculating the gradient of the objective function. This gradient calculation is analogous to an N-body problem, requiring summing the contributions between the $O(N^2)$ pairs of points. Naive implementation of quadratic algorithms do not scale gracefully to large N. In low dimensions, such as the case we target with projecting to 2 dimensions, N-Body approximation strategies exist which substantially reduce the complexity of such calculations down to $O(N \log N)$ using the quadtrees or dual-trees, or even $O(N)$ with the fast multipole method [47]. De Freitas et al propose using the Dual-Tree method[48] to approximate the SNE gradient. In contrast, quadtrees are used by both van der Maaten for t-SNE [4] and Yang [28] for NE. Van der Maaten showed that, with respect to the performance of optimizing the t-SNE objective function, the need to reconstruct the dual tree at each iteration reduces the computational advantages of the Dual Tree approach over the quadtree [4]. Following their lead, we also chose to use quadtrees to approximate the gradient of the t-SNE objective function.

An important parameter in N-Body approximation strategies with quadtrees is the accuracy parameter θ. In their parametric analysis of θ with respect speed and accuracy of tSNE layout results, van der Maaten highlight a stable region between 0.25 and 0.75 and select 0.5 as a good tradeoff in speed versus quality [4]. We follow suit and set θ equal to 0.5 in all our results below.

5. Experiments

We now report on the results of experiments that compare the performance of Q-SNE to other approaches. We first describe the benchmark datasets we use in the experiments. We evaluate the performance of the approximate NN component

Dataset	**Docs (N)**	**Terms (M)**
`warlogs`	3,077	4,337
`metacombine`	28,433	28,377
`textfiles`	41,591	1,645,053
`cables`	251,287	771,854
`nytimes`	299,752	102,660
`pubmed`	1,999,000	141,043

Table 1: The six benchmark datasets used in the experiments, with document and term counts.

on document data by comparing it to three alternatives, measuring both computational speed with runtimes and quality in terms of precision-recall curves. We then evaluate the performance of the layout component, again with quantitative measures of speed and quality, against five alternatives. Finally, we perform a detailed comparison of BH-SNE and Q-SNE results where qualitative analysis is linked to quantitative measurements, to explain what aspects of visible structure in the scatterplots that are misleading with BH-SNE are more trustworthy with Q-SNE. All benchmarks were run on the same evaluation computer, a Dell R720 Server with two 12-core 2GHz Xeon CPUs and 32GB of RAM.

5.1. Benchmark Datasets

The benchmarks in this paper are run on a set of six text datasets, all gathered from real-world sources. The `warlogs` and `cables` datasets are sets of documents from Wikileaks, where `warlogs` represents the July 2009 subset of the Iraqi Warlogs, and `cables` contains the entire set of text files from the Cablegate dataset. The `metacombine` dataset is produced from metadata tags harvested from a disparate collection of digital libraries [49]; we ran IDF as a post-processing step, because the original had only been processed with TF. The `textfiles` dataset is harvested from a curated collection of 1980s-era bulletin board system posts[1]. We gathered `nytimes`, New York times news articles, and `pubmed`, PubMed abstracts, from the Bag of Words dataset from the UCI Machine Learning Repository [50]. Table 1 summarizes the sizes of each collection in terms of the number of documents (points) N and number of terms (dimensions) M.

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

5.2. Approximate Nearest Neighbor Results

We first report on experiments that justify our selection of the APQ inverted-file based approximate nearest neighbor algorithm. We use the same C implementation of APQ that was built during its original development [3]. We compare APQ to three state-of-the-art algorithms. Two are representatives from each of the other two major families of NN algorithms. We select the Vantage-Point metric tree (VPTree) algorithm from the spatial partitioning family [5]. We use the C++ VPTree implementation included in the BH-SNE source distribution.[2] We select Locality Sensitive Hashing (LSH) from the mapping family [37]. We use the C++ implementation from the Caltech Large Scale Image Search Toolbox,[3] after modifying it to handle sparse document vectors for greater scalability, with the parameter settings of 10 hash tables and 10 random-projection hash functions per table. The third algorithm tested is PCA-VPTree, the hybrid NN method used in BH-SNE, where the document matrix is first reduced to 50 dimensions with PCA and then the VPTree algorithm is applied to the results. The BH-SNE source distribution does not include code for the reduction step at all, but assumes that input data has already been preprocessed. In order to compare PCA-VPTree to the alternatives and to compare BH-SNE to Q-SNE on datasets with more than 50 terms, we used an existing fast SVD approximation [12] in GNU Octave to compute PCA on each data set. This implementation can only handle matrices of up to 2 GB before running out of address space, but the 2 million document `pubmed` dataset requires more storage than this limit and could not be processed. Datasets of this size could possibly be handled with existing sparse PCA approaches, for example by building on a toolkit capable of large matrix storage such as scipy [51], but we leave this engineering effort to future work.

Figure 5.2 shows the effect of collection size on the speeds of the tested NN algorithms with a log-log plot. VPTree is the slowest method tested, taking nearly one day for the 251K case. For the smaller datasets of 3K and 28K size, LSH and PCA-VPTree were slightly faster than APQ. For the larger datasets of 50K and beyond, APQ is substantially faster than the others. It is the only algorithm capable of handling millions of documents; the LSH, PCA, and VPTree implementations could not handle these sizes due to excessive memory consumption.

We measure the quality of the NN results using precision-recall (PR) curves, as discussed in Section 3. Figure 5.2 shows the mean PR curves for the 30 nearest

[2]http://homepage.tudelft.nl/19j49/t-SNE_files/bh_tsne.tar.gz

[3]http://vision.caltech.edu/malaa/software/research/image-search/

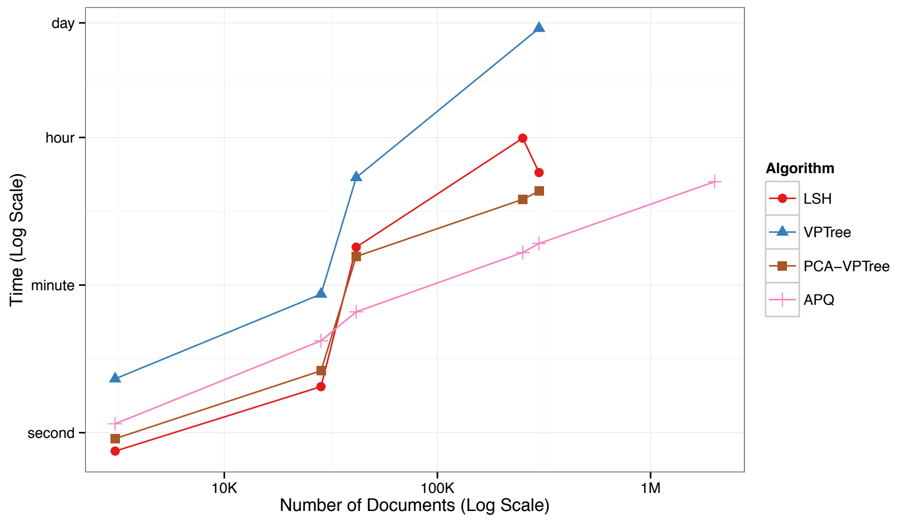

Figure 1: Log-Log plot of NN algorithm runtimes against dataset size, for computing the 30 approximate nearest neighbors. APQ (pink curve) is substantially faster than LSH, VPTree, or the hybrid PCA-VPTree for large datasets.

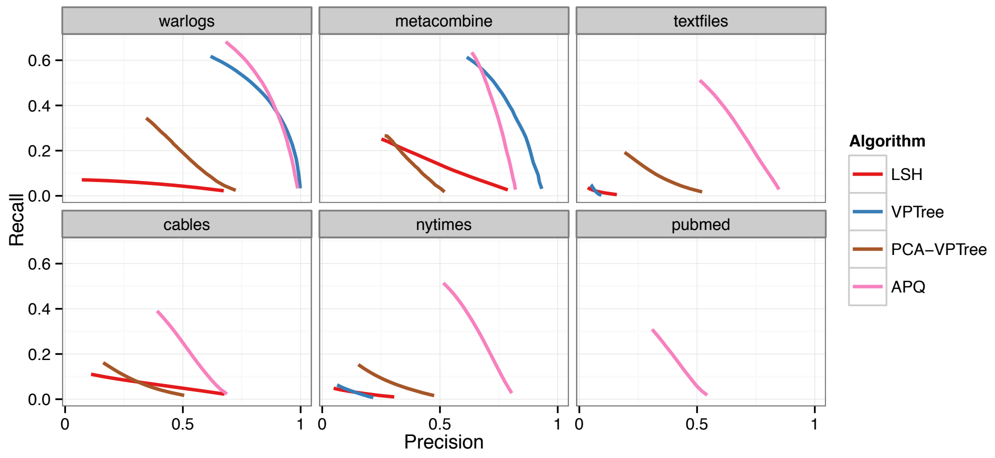

Figure 2: Precision-recall curves comparing the quality of the approximation for the 30 nearest neighbors across the four algorithms. APQ (pink curve) is substantially better than the alternatives for all of the datasets, with the exception of the tiny `warlogs` dataset where VPTree matches its performance. Top left: `warlogs` (N=3K). Top middle: `metacombine` (N=28K). Top right: `textfiles` (N=42K). Bottom left: `cables` (N=251K). Bottom middle: `nytimes` (N=300K). Bottom right: `pubmed` (N=2M).

neighbors computed by each of the four NN approaches that we tested. To calculate each curve, we fix $r = 30$ and iterate k from 1 to 30, calculate the number of true positives t in the k nearest neighbors, set precision to t/k and recall to t/r for each k, and then draw a line connecting each point. Because calculating the true nearest neighbors for the larger collections is extremely costly, we sample 100K documents uniformly at random and take the mean of the sampled precision and recall.

For the tiny (N=3K) `warlogs` and smaller `metacombine` datasets, the APQ and VPTree results are competitive with each other, and superior to the others. For all of four of the larger datasets, APQ yields high-fidelity results, performing substantially better than the other three methods. It is the only method that could handle the largest `pubmed` dataset, as we discuss above, so we cannot make statements about relative quality; the absolute quality is reasonable.

These experiments validate our claims that APQ scales well for large document collections, providing substantial speed and quality improvements over the alternative approaches to finding approximate nearest neighbors.

In the case where the input distances are derived from a true metric, VPTree will return a perfect precision and recall. For PCA-VPTree, we employ metric euclidean distances to compute nearest neighbors after performing PCA. Thus, the precision-recall curves of PCA-VPTree measure the corruption of point neighborhoods induced by the reduction to 50 dimensions by PCA.

5.3. Layout Results

We compare the layout performance of Q-SNE to five alternative DR algorithms. We choose the popular Principal Component Analysis (PCA) [7] approach as a baseline that is fast but known to be of low quality; it is in the orthogonal projection family of DR algorithms. We choose the Glimmer [8] MDS algorithm as a representative global distance method, where the optimization criterion is stress. We include two hybrid methods that optimize a combination of stress and projection: Local Affine Multidimensional Projection (LAMP) [10] using the Projection Analyzer tool[4], and Least Squared Projection (LSP) [9] using the Projection Explorer [52]; the latter was specifically designed for document data. For input parameters, we use the defaults proposed by the authors of the original systems. In the case of LSP, the number of control points is $N/10$, number of nearest neighbors is 15, and force improvement iterations are 50. For LAMP we set the number

[4]http://code.google.com/p/projection-analyzer/

of control points to $3\sqrt{N}$ uniformly chosen points at random. Finally, we compare against Barnes-Hut Stochastic Neighborhood Embedding (BH-SNE) [4], as the most similar method to Q-SNE; both use Kullback-Leibler Divergence as the optimization criterion. We run both BH-SNE and Q-SNE for 5000 iterations of gradient descent with 2000 high-energy iterations and a perplexity of 10.

As in the previous section, we compare against the same six benchmark datasets with respect to runtime speeds and mean PR curves. We include a set of scatterplot layouts to illustrate the qualitative differences between the results generated by the different algorithms.

Figure 3 shows the time needed to compute layouts by the different DR algorithms, again with a log-log plot of runtimes against dataset size. PCA shines in terms of runtimes compared to the other four, thanks to very efficient and accurate SVD approximation algorithms; however, as shown below, there is a speed-accuracy tradeoff: its quality is very low. The other four DR algorithms tested exhibit roughly commensurate computation speeds. The BH-SNE and Q-SNE performance numbers both include the nearest-neighbor calculation times, for fair comparison against the other algorithms that do not have separable phases where the equivalent computation is done. We do not include document processing times for either algorithm. The BH-SNE and Q-SNE times are so similar that their performance curves are directly on top of each other, thus showing that total computation time is dominated by the layout phase rather than the NN computation phase. We halted the Glimmer computation on `textfiles` when it failed to terminate after over 30 hours, and the implementation did not scale to the three larger datasets. Both of the freely available LSP and LAMP implementations were unable to handle these four large datasets at all. There are no results for BH-SNE on `pubmed` because of the preprocessing limitations discussed in Section 5.2.

We quantitatively measure the layout quality with mean PR curves, following the method of Venna et al [6] by computing precision and recall based on the local neighborhoods and comparing their content to the true neighborhoods. Figure 4 shows the resulting PR curves for the six benchmark datasets.

The results largely mirror those for the NN computations, validating our hypothesis that improving the nearest-neighbor computation would improve the final layout. Q-SNE yields the best PR curves on all datasets by a large margin, except on the tiny `warlogs` dataset where BH-SNE roughly matches its performance. For the four larger datasets, the quality of BH-SNE was substantially lower than Q-SNE and substantially higher than PCA. As expected, the quality of the PCA layouts is very low, showing that PCA is not a viable alternative for document data despite its speed.

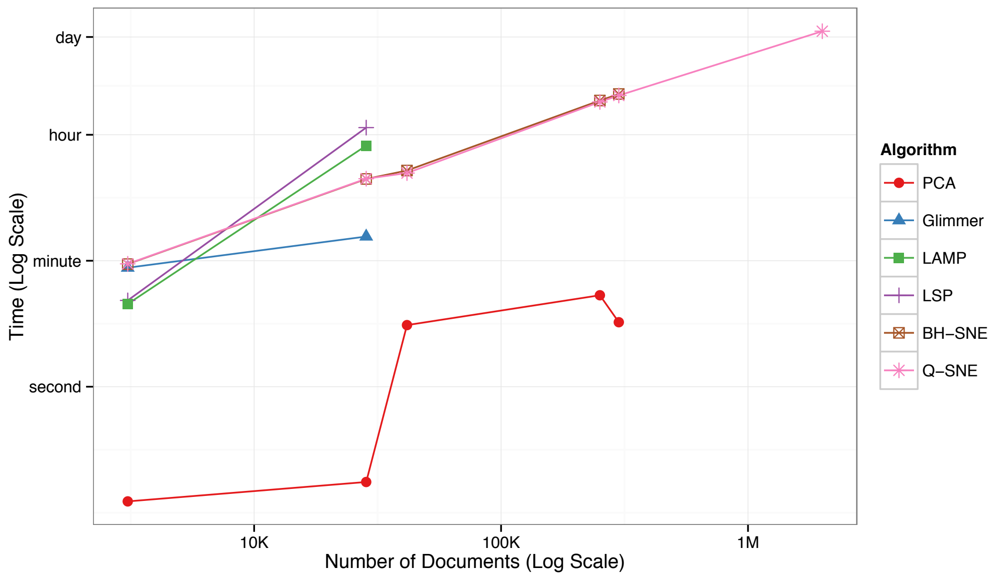

Figure 3: Log-Log plot of the time to complete the full DR layout computation as a function of the number of documents in the dataset. The BH-SNE (brown) and Q-SNE (pink) timings are identical for the datasets that both can handle. PCA (red) is fast, but poor quality.

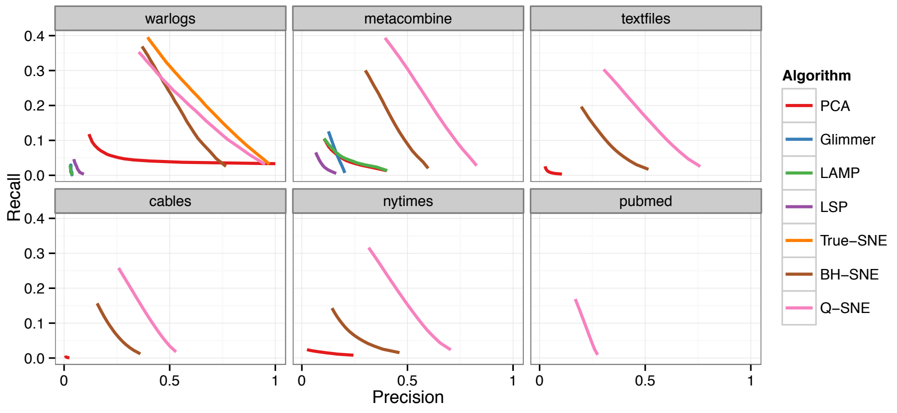

Figure 4: Precision-recall curves comparing for DR layouts across the six algorithms. Q-SNE (pink) yields substantially better quality than the alternatives, except for the tiny `warlogs` dataset where BH-SNE (brown) nearly matches its performance. Curves are absent for algorithms that could not scale to the larger datasets; for these, BH-SNE yields intermediate quality, substantially better than the low-quality PCA performance (red). Top left: `warlogs` (N=3K). Top middle: `metacombine` (N=28K). Top right: `textfiles` (N=42K). Bottom left: `cables` (N=251K). Bottom middle: `nytimes` (N=300K). Bottom right: `pubmed` (N=2M).

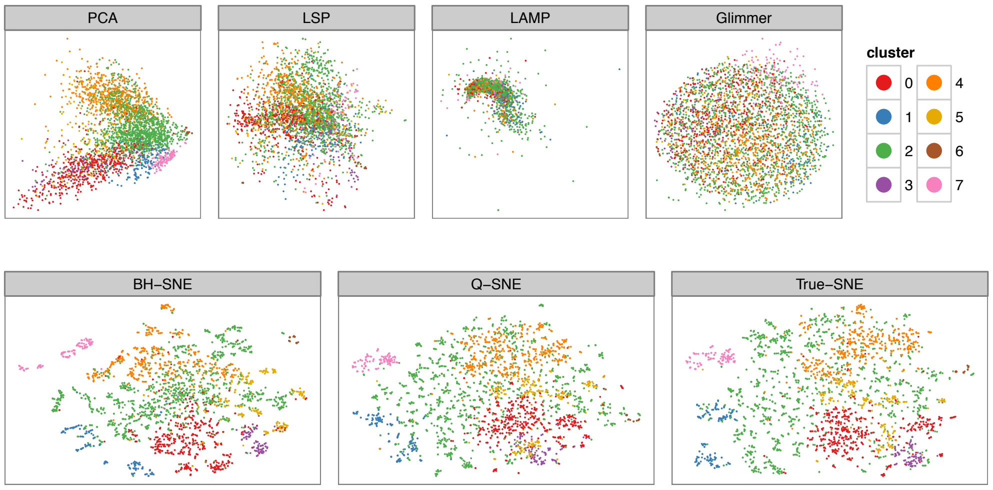

Figure 5: 2D layouts for the tiny `warlogs` (N=3K) dataset for all six DR algorithms plus a true baseline tested with cluster labeling from Ward's method (k=8). The qualitative results match the quantitative analysis with PR curves in Figure 4: the layout quality for PCA, LSP, LAMP, and Glimmmer on the top row is poor, with no visible cluster structure, as opposed to the higher-quality layouts for BH-SNE, Q-SNE, and True-SNE which correspond well to derived clusters.

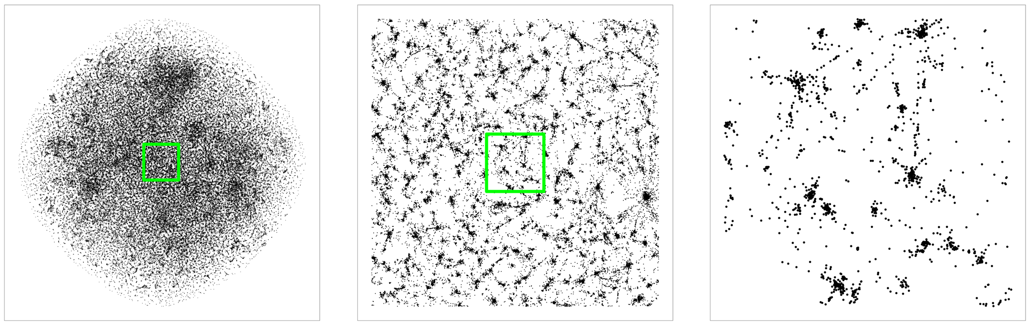

Figure 6: Q-SNE layout of the `pubmed` dataset (N=2M). Left: Overview. Middle: Partial zoom. Right: Full zoom. The layout contains some large-scale cluster structure visible in the overview, and many smaller clusters that are hard to distinguish at that scale. The cluster structure is more evident after zooming.

Figure 5 shows scatterplots for the tiny warlogs dataset for all six layout algorithms plus a reference benchmark we call True-SNE. True-SNE minimizes the same energy function as BH-SNE and Q-SNE, but uses the true set of nearest neighbors and their distances, not an approximation. The qualitative results match quantitative results shown in the PR curves in Figure 4: the four algorithms of PCA, LSP, LAMP and Glimmer all fail to provide clear separation between clusters and have lower precision and recall, while BH-SNE, Q-SNE, and True-SNE all do resolve visible cluster structure and report higher precision and recall. Because calculating the true nearest neighbors does not scale well with the size of the data, we only include a precision-recall curve for True-SNE on the warlogs dataset. For reference, we color points in each scatterplot with 8 cluster labels derived using Ward's agglomerative clustering method [53]. In contrast to the other layout methods, the visually distinct modes of point distributions in different SNE layouts correlate well with clusters derived from the local connectivity of the original data.

Figure 6 shows the Q-SNE layout of the very large (N=2M) pubmed dataset, at three levels of zoom. The overview on the left shows some large-scale cluster structure, but many smaller features. We note that these features are not individual points, but rather entire small-scale clusters. This cluster structure is more apparently in the partial zoom in the middle. In the fully zoomed detail view on the right, the smallest visible features are indeed individual documents, and the cluster structure is very clear.

The obvious question is whether the lack of mid-level structure in the layout is a true negative result, indicating that such structure simply does not exist in this very large collection of individual paper abstracts, or is a false negative result where Q-SNE failed tease out existing structure. We cannot directly compare this layout to alternatives because of the limitations of their implementations and/or algorithms. The lone PR curve for this layout in Figure 4 is equivocal: it is substantially lower than for the smaller datasets, hinting that some aspects of dataset structure have not been recovered. We call for future work to shed further light on the true structure of this dataset.

Further scatterplots appear in Figures 7 and 8, which display side-by-side comparisons of BH-SNE and Q-SNE layouts of the (N=42K) textfiles and (N=251K) cables datasets respectively; in the next section we compare these results in more detail as part of our analysis of how these two algorithms differ.

5.4. Q-SNE and BH-SNE

The different choices of approximate nearest neighbor techniques between BH-SNE and Q-SNE algorithms lead to both quantitative differences in terms of PR curves and qualitative differences in terms of the visual character of their layouts. We now perform an in-depth comparison of these layout results of these two approaches to better understand the qualitative differences of the two different techniques.

The two columns in Figure 7 show layouts from the medium (N=42K) `textfiles` dataset, with BH-SNE on the left and Q-SNE on the right. Because this dataset has been hand classified by a human curator, we have ground-truth categorization of the data points, and use that to color the points. Each row shows 7 of the categories with color, with all other points in a neutral black. We use this small-multiple approach because of the strong limits on the number of colors that readers can distinguish. The three rows show 21 of the 28 categories in the dataset; the others follow roughly the same pattern. We see that both algorithms separate points into point densities with spatial location correlating well with the ground-truth categorization. However, the Q-SNE layout tends to break down larger categories into a diffuse set of smaller groupings, as is evident in the purple `sex` category in the bottom row.

Figure 8 shows layouts from the large (N=251K) `cables` dataset, with BH-SNE and Q-SNE shown side by side. As in the case of the `textfiles` dataset, both algorithms produce layouts with varying point densities; that is, there are obvious clustered regions of points in both cases. The layouts produced by BH-SNE have fewer clusters that are both larger and tighter, with very visible separations between high density regions. The Q-SNE layouts have more clusters that are smaller, with more points that fall into the background between any visibly distinguished cluster. People conducting visual analysis of DR data rely strongly on visual cluster separation when making judgements about dataset structure [1]. The crisp separation in the BH-SNE layout gives the strong visual impression of meaningful cluster structure in the dataset. However, the quantitative metrics shown in the PR curves imply that the Q-SNE layout is a more accurate depiction of true dataset structure. We conjecture that some of the strong cluster structure shown in the BH-SNE layout is a misleading artifact, possibly due to the loss of important information by the initial reduction to 50 dimensions within the approximate NN calculation. We now check this conjecture with more detailed analysis.

Figure 9 displays two kernel density estimates of the entire set of recalls for the `cables` dataset with the setting $k = 15$ and $r = 30$. We observe that the Q-SNE plot shows its estimated density more uniformly distributed in the higher-recall

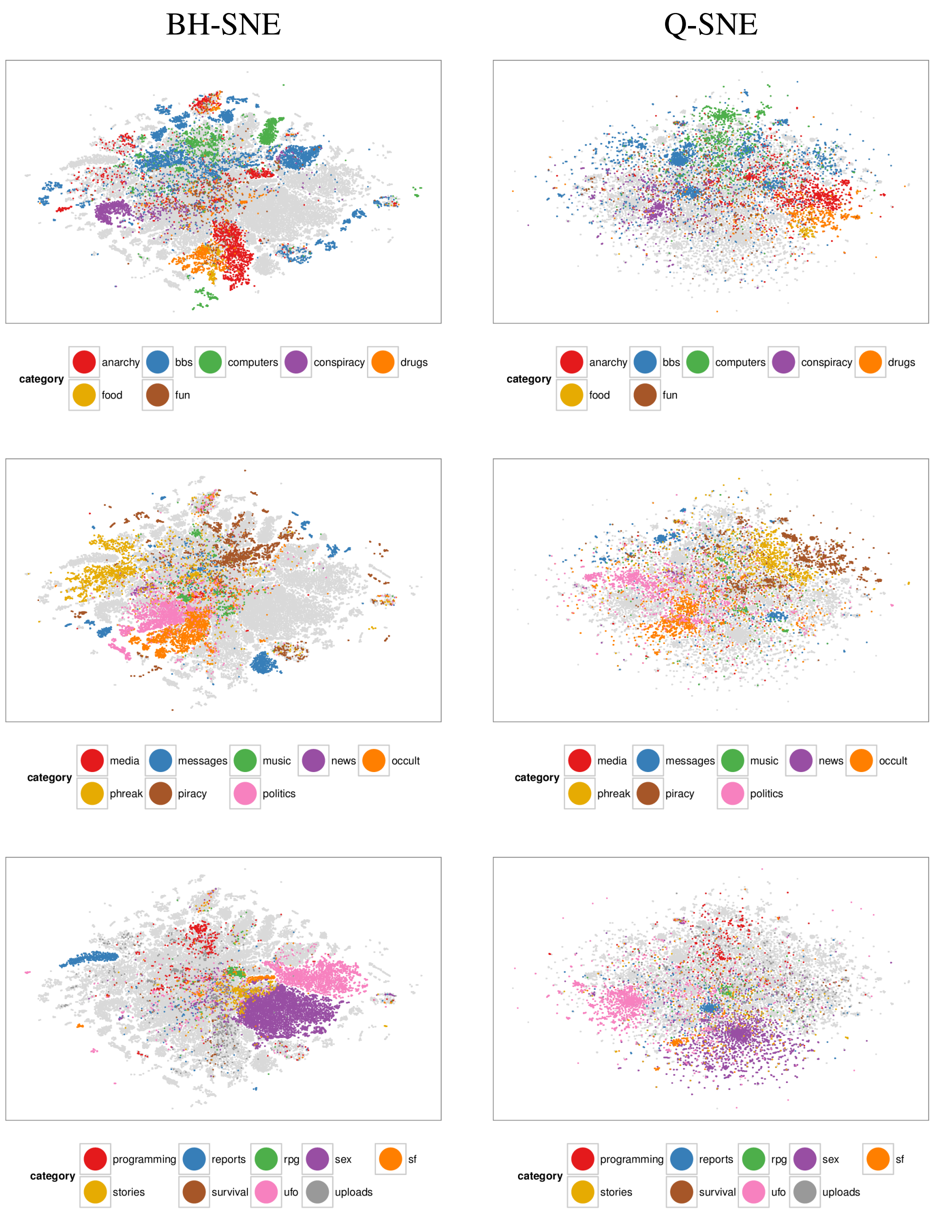

Figure 7: Layouts of the `textfiles` (N=42K) dataset, colored according to manually categorized ground truth. Left column: BH-SNE. Right column: Q-SNE. Visual document clusters correspond well to the given ground truth for both.

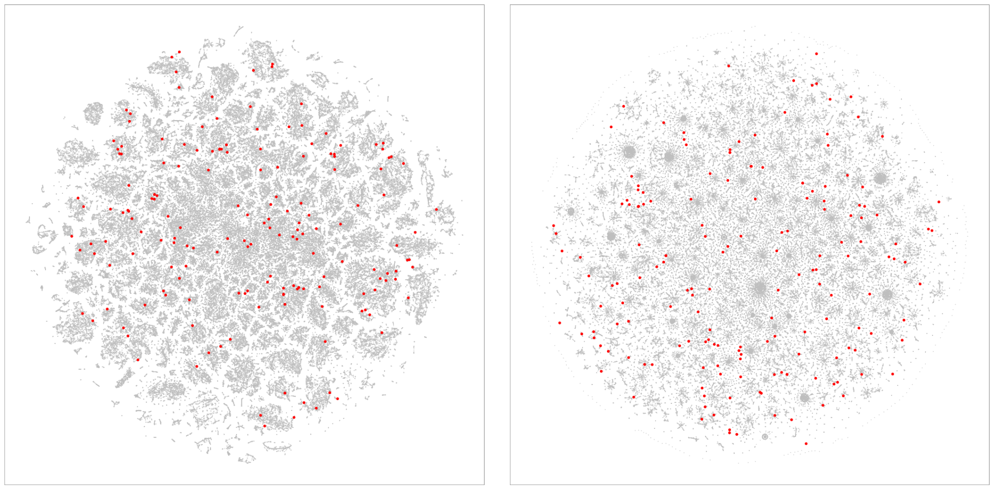

Figure 8: 2D layouts for the `cables` (N=251K) dataset, with red dots marking the location of samples that had higher precision in Q-SNE than BH-SNE. Left: BH-SNE. Right: Q-SNE.

regions of the plot. It is instructive to analyze the locations of samples in the Q-SNE plot that possess higher precision than the same BH-SNE samples, shown as red dots in Figure 8. We observe that most of the higher recall points are located in low-density regions of the Q-SNE plot. In the BH-SNE plot we do not observe an obvious correlation between plot density and lower precision; instead, the red dots are placed near the edges of the visible clusters. Both of these phenomena provide supporting evidence for our conjecture that the BH-SNE layout has too few clusters. Q-SNE lays out these red points between clusters, showing that they have true neighbors across multiple clusters. BH-SNE lays out these red points within clusters, but near the periphery: the points are pushed away from the center because they do not have close relationships with most of the points in the cluster, but enough of their neighbors are placed within the same large cluster that they stay within it.

The red samples in Figure 8 with higher precision will all, by definition, have a higher proportion of true nearest neighbors in the layout. To illustrate the visual effect of higher precision, we select a random point from the red set of higher Q-SNE recall samples and analyze the difference in true neighborhood distribution between the Q-SNE and BH-SNE plots, by highlighting the selected point's 99 true nearest neighbors in blue. Figure 10 displays these neighborhoods at three different zoom levels for both plots, with the BH-SNE layouts on the left and Q-SNE layouts on the right. At the overview level on the top, we highlight all 99 true

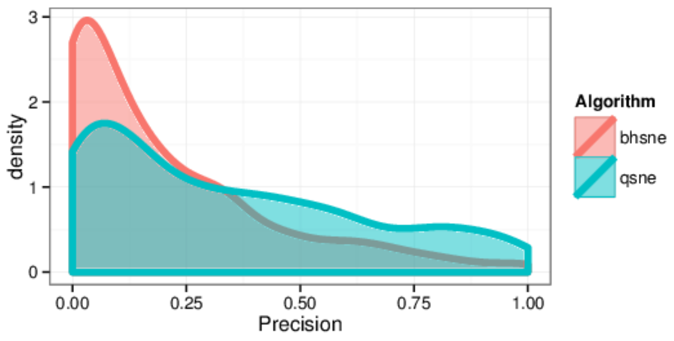

Figure 9: Kernel density estimates comparing precision values for BH-SNE and Q-SNE, for the `cables` datasets with the $k = 15$ retrieved documents and $r = 30$ relevant documents.

NN points in blue. We can see that most of them are in the vicinity of the red point on both plots, with a few more visible and distant outliers for BH-SNE. The next row is zoomed further in, and only the 20 closest true NN points are highlighted in blue. We can observe a very clear difference in the distribution of neighborhoods. In the BH-SNE plot, the blue neighboring points all fall within a single large cluster that is visibly separated from others, but they are fairly spread out across it. In the Q-SNE plot, the blue points are so tightly packed into a small cluster around the red selected point that they are barely visible behind it. The range shown in each zoomed-in plot is calibrated between both sides, so that the cluster sizes are directly comparable. The highest level of zoom, at bottom, again shows dramatic differences. The BH-SNE plot shows only loose clustering, and only three points are visible; the rest are outside the visible region. The Q-SNE plot shows a very tight concentration of around 10 blue points around the red one; the rest are not visible because they underneath the other plotted points, not because they are outside the bounds of the window. This analysis shows that the more fine-grained cluster structure of Q-SNE layouts does reflect true local structure more accurately than the coarse-grained structure of BH-SNE layouts. This figure shows a representative example; our investigation shows that this difference in the distribution of true neighbors holds for most of the higher-recall points shown in red in Figure 8.

We conclude that the placement of documents into large clusters with clean visual groupings by BH-SNE is indeed a misleading artifact. The messier visible structure in the Q-SNE plots is a layout where gaps between medium-sized clusters are filled in with smaller clusters. Our analysis connects the improvements in

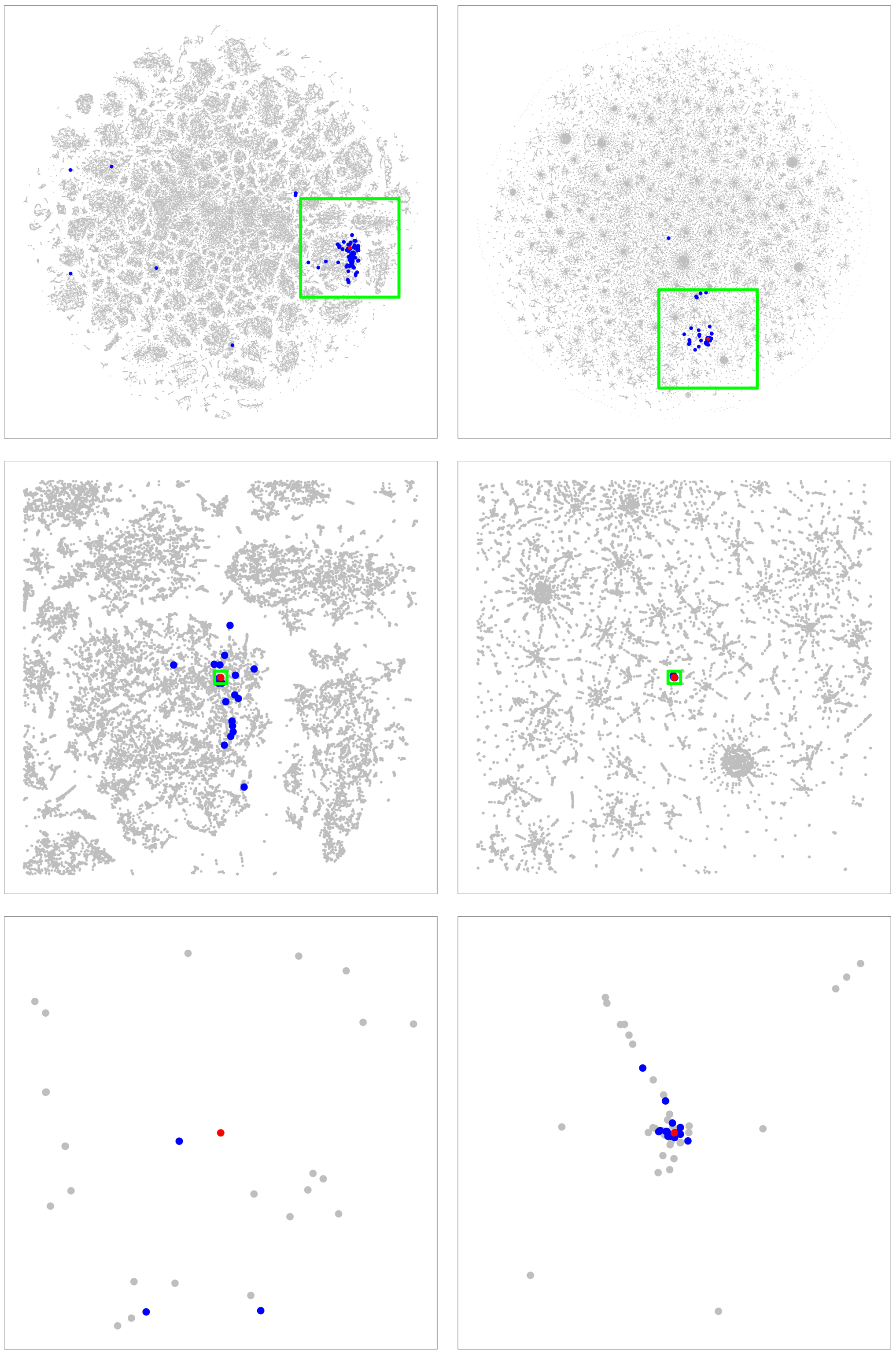
Figure 10: Analyzing nearest neighbors (blue) for selected point (red), for `cables`. Left column: BH-SNE. Right column: Q-SNE. Top: overview, with 99 NN. Middle row: partial zoom, with closest 20 NN. Bottom row: full zoom.

mean precision and recall shown in the quantitative analysis from the PR curves with a qualitative understanding of how the layout better represents the actual neighborhood structure of the data.

6. Conclusion and Future Work

We have presented the Q-SNE technique for the dimensionality reduction of large document collections. Q-SNE builds on the BH-SNE [4] technique, using the same Barnes-Hut approximations for efficiently compute the gradient when minimizing the t-SNE objective function during the layout itself. The major difference is the approach to computing the approximate k-nearest neighbors. Q-SNE exploits the query-based APQ algorithm that uses impact-ordered inverted files [3], rather than the hybrid combination of reducing to 50 dimensions using PCA and then running Vantage Point trees [5]. Our hypothesis that this more sophisticated approach to NN computation would yield higher-quality layouts was validated by extensive experimental results. We first showed that inverted files outperform 3 other nearest neighbor techniques on 6 large document datasets in terms of both runtimes and mean precision and recall. We then showed that our approach to dimensionality reduction outperformed 5 other techniques in terms of the combination of runtimes and quality as measured by mean precision and recall. We also present evidence for the benefits of our approach through a qualitative discussion of the layout scatterplots across these DR techniques. Finally, we performed an in-depth comparison of the results produced by the BH-SNE and Q-SNE methods on two benchmark datasets, illuminating differences between the two methods both qualitatively and quantitatively.

Future work to improve the Q-SNE algorithm itself includes improving the optimization framework to have automatic termination criterion that detects convergence, rather than simply using a fixed number of iterations. It would also be useful to select the number of nearest neighbors k automatically, rather than relying on the user to choose an appropriate value. An obvious improvement from an interactive visualization point of view would be to better support exploratory visual analysis of the scatterplots would be to add browsing capability with an interactive analytics tool that would allow easy selection and inspection of visual clusters. Another obvious improvement from an IR point of view would be to use more sophisticated document scoring methods than TF-IDF, such as BM25F[54].

In the larger picture, it may be useful and straightforward to adapt the t-NERV [6] algorithm to incorporate query-based NN computation, since NERV is a generalization of SNE. A more challenging project would be to develop a mathematical framework that handles a preponderance of unit-length distances gracefully without using probabilities.

References

[1] M. Sedlmair, T. Munzner, M. Tory, Empirical guidance on scatterplot and dimension reduction technique choices, IEEE Trans. on Visualization and Computer Graphics (Proc. InfoVis) 19 (12) (2013) 2634–43.

[2] M. Tory, D. Sprague, F. Wu, W. Y. So, T. Munzner, Spatialization design: comparing points and landscapes., IEEE Trans. on Visualization and Computer Graphics (Proc. InfoVis) 13 (6) (2007) 1262–9.

[3] S. Ingram, Practical considerations for dimensionality reduction: User guidance, costly distances, and document data, Ph.D. thesis, University of British Columbia, Department of Computer Science, chapter 4 (2013).

[4] L. van der Maaten, Barnes-Hut-SNE, in: Proc. Intl. Conf. Learning Representations (ICLR), 2013, arXiv:1301.3342.

[5] P. N. Yianilos, Data structures and algorithms for nearest neighbor search in general metric spaces, in: Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA), 1993, pp. 311–321.

[6] J. Venna, J. Peltonen, K. Nybo, H. Aidos, S. Kaski, Information retrieval perspective to nonlinear dimensionality reduction for data visualization, Journal of Machine Learning Research 11 (2010) 451–490.

[7] I. T. Jolliffe, Principal Component Analysis, 2nd Edition, Springer, 2002.

[8] S. Ingram, T. Munzner, M. Olano, Glimmer: Multilevel MDS on the GPU, IEEE Trans. Visualization and Computer Graphics (TVCG) 15 (2) (2009) 249–261.

[9] F. Paulovich, L. Nonato, R. Minghim, H. Levkowitz, Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping, IEEE. Trans. Visualization and Computer Graphics (TVCG) 14 (3) (2008) 564–575.

[10] P. Joia, F. V. Paulovich, D. Coimbra, J. A. Cuminato, L. G. Nonato, Local affine multidimensional projection, IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis) 17 (2011) 2563–71.

[11] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Information processing & management 24 (5) (1988) 513–523.

[12] N. Halko, P.-G. Martinsson, Y. Shkolnisky, M. Tygert, An algorithm for the principal component analysis of large data sets, SIAM Journal on Scientific Computing 33 (5) (2011) 2580–2594.

[13] I. Borg, P. Groenen, Modern multidimensional scaling: Theory and applications, Springer, 2005.

[14] S. L. France, J. Carroll, Two-way multidimensional scaling: A review, IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews 41 (5) (2011) 644–661.

[15] A. Buja, D. F. Swayne, Visualization methodology for multidimensional scaling, Journal of Classification 19 (1) (2002) 7–43.

[16] J. W. Sammon Jr, A nonlinear mapping for data structure analysis, IEEE. Trans. Computers 100 (5) (1969) 401–409.

[17] J. Tenenbaum, V. de Silva, J. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.

[18] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.

[19] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, Advances in Neural Information Processing Systems 14 (2001) 585–591.

[20] V. Silva, J. Tenenbaum, Global versus local methods in nonlinear dimensionality reduction, Advances in Neural Information Processing Systems 15 (2003) 705–712.

[21] K. Weinberger, L. Saul, An introduction to nonlinear dimensionality reduction by maximum variance unfolding, in: Proc. Natl. Conf. Artificial Intelligence (AAAI), Vol. 21, 2006, pp. 1683–1686.

[22] N. Lawrence, Probabilistic non-linear principal component analysis with Gaussian process latent variable models, Journal of Machine Learning Research 6 (2005) 1783–1816.

[23] L. Chen, A. Buja, Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis, Journal of the American Statistical Association 104 (485) (2009) 209–219.

[24] M. Sedlmair, M. Brehmer, S. Ingram, T. Munzner, Dimensionality reduction in the wild: Gaps and guidance, Tech. rep., Dept. of Computer Science, University of British Columbia (2012).

[25] L. Van der Maaten, E. Postma, H. Van Den Herik, Dimensionality reduction: A comparative review, Journal of Machine Learning Research (JMLR) 10 (2009) 1–41.

[26] G. Hinton, S. Roweis, Stochastic neighbor embedding, Advances in Neural Information Processing Systems (NIPS) 15 (2002) 833–840.

[27] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, Journal of Machine Learning Research 9 (2579-2605) (2008) 85.

[28] Z. Yang, J. Peltonen, S. Kaski, Scalable optimization of neighbor embedding for visualization, Journal of Machine Learning Research 28 (2) (2013) 127–135.

[29] R. Minghim, F. V. Paulovich, A. A. Lopes, Content-based text mapping using multi-dimensional projections for exploration of document collections, in: Proc. Visualization and Data Analysis (VDA), Vol. 6060, SPIE Press, 2006, p. 60600S.

[30] C. Faloutsos, K.-I. Lin, FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets, Vol. 24, ACM, 1995.

[31] L. Rousseeuw, L. Kaufman, Clustering by means of medoids, Statistical Data Analysis Based on the L1-norm and Related Methods (1987) 405–416.

[32] F. V. Paulovich, R. Minghim, Hipp: A novel hierarchical point placement strategy and its application to the exploration of document collections, Visualization and Computer Graphics, IEEE Transactions on 14 (6) (2008) 1229–1236.

[33] C. D. Manning, P. Raghavan, H. Schütze, Introduction to information retrieval, Cambridge University Press, 2008.

[34] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, A. Y. Wu, An optimal algorithm for approximate nearest neighbor searching fixed dimensions, Journal of the ACM (JACM) 45 (6) (1998) 891–923.

[35] J. L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (9) (1975) 509–517.

[36] A. Beygelzimer, S. Kakade, J. Langford, Cover trees for nearest neighbor, in: Proc. Intl. Conf. Machine Learning (ICML), ACM, 2006, pp. 97–104.

[37] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: Proc. Intl. Conf. on Very Large Data Bases (VLDB), 1999, pp. 518–529.

[38] J. Zobel, A. Moffat, Inverted files for text search engines, ACM Computing Surveys (CSUR) 38 (2) (2006) 6.

[39] F. Murtagh, A very fast, exact nearest neighbor algorithm for use in information retrieval, Information Technology: Research and Development 1 (4) (1982) 275–283.

[40] A. F. Smeaton, C. Van Rijsbergen, The nearest neighbour problem in information retrieval: an algorithm using upperbounds, ACM SIGIR Forum 16 (1) (1981) 83–87.

[41] R. J. Bayardo, Y. Ma, R. Srikant, Scaling up all pairs similarity search, in: Proc. Intl. Conf. World Wide Web (WWW), 2007, pp. 131–140.

[42] C. Xiao, W. Wang, X. Lin, H. Shang, Top-k set similarity joins, in: IEEE Conf. Data Engineering (ICDE), 2009, pp. 916–927.

[43] V. N. Anh, A. Moffat, Impact transformation: effective and efficient web retrieval, in: Proc. ACM Conf. Information Retrieval (SIGIR), 2002, pp. 3–10.

[44] C. Böhm, The similarity join: A powerful database primitive for high performance data mining, in: IEEE Conf. on Data Engineering (ICDE) Tutorial, 2001.

[45] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. V. der Vorst, Templates for the Solution of

Linear Systems: Building Blocks for Iterative Methods, 2nd Edition, SIAM, 1994.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[47] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, Journal of computational physics 73 (2) (1987) 325–348.

[48] N. de Freitas, Y. Wang, M. Mahdaviani, D. Lang, Fast krylov methods for n-body learning, in: Advances in Neural Information Processing Systems, 2005, pp. 251–258.

[49] A. Krowne, M. Halbert, An initial evaluation of automated organization for digital library browsing, in: Proc. ACM/IEEE Joint Conf. Digital Libraries (JCDL), IEEE, 2005, pp. 246–255.

[50] K. Bache, M. Lichman, UCI machine learning repository (2013).
URL `http://archive.ics.uci.edu/ml`

[51] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python (2001–).
URL `http://www.scipy.org/`

[52] F. V. Paulovich, M. C. F. Oliveira, R. Minghim, The projection explorer: A flexible tool for projection-based multidimensional visualization, in: Proc. Brazilian Symp. Computer Graphics and Image Processing (SIBGRAPI), IEEE CS Press, 2007, pp. 27–36.

[53] J. H. Ward Jr, Hierarchical grouping to optimize an objective function, Journal of the American Statistical Association 58 (301) (1963) 236–244.

[54] H. Zaragoza, N. Craswell, M. J. Taylor, S. Saria, S. E. Robertson, Microsoft Cambridge at TREC 13: Web and hard tracks., in: TREC, Vol. 4, 2004, pp. 1–1.