TWO-DIMENSIONAL VISUALIZATION OF LARGE DOCUMENT LIBRARIES USING T-SNE

Rita González-Márquez, Philipp Berens, Dmitry Kobak
University of Tübingen, Germany
{rita.gonzalez-marquez, philipp.berens, dmitry.kobak}@uni-tuebingen.de

ABSTRACT

We benchmarked different approaches for creating 2D visualizations of large document libraries, using the MEDLINE (PubMed) database of the entire biomedical literature as a use case (19 million scientific papers). Our optimal pipeline is based on log-scaled TF-IDF representation of the abstract text, SVD preprocessing, and t-SNE with uniform affinities, early exaggeration annealing, and extended optimization. The resulting embedding distorts local neighborhoods but shows meaningful organization and rich structure on the level of narrow academic fields.

1 INTRODUCTION

Many academic fields that traditionally did not have to deal with high-dimensional data analysis now face the challenge of having rich datasets with millions of samples. One example is digital humanities, with digital libraries now providing millions of documents. Here we focus on two-dimensional visualization of such libraries based on the document text (Schmidt, 2018). An alternative approach is to base visualization on the citation graph (Noichl, 2021) (see also `opensyllabus.org`, `paperscape.org`, or `connectedpapers.com`), which is beyond the scope of our analysis.

One of the most popular dimensionality reduction algorithms for 2D visualization is t-SNE (van der Maaten & Hinton, 2008), based on the earlier SNE (Hinton & Roweis, 2002). Modern implementations (Linderman et al., 2019; Artemenkov & Panov, 2020) allow to run t-SNE on millions of samples. Closely related algorithms include LargeVis (Tang et al., 2016) and UMAP (McInnes et al., 2018). These methods have already been used to visualize document corpora such as the HathiTrust library ($n = 14$ M) (Schmidt, 2018) or its subsets (Kobak et al., 2019), but there has been no systematic benchmark of different algorithms and processing choices.

Here we develop a t-SNE pipeline to produce a visualization (Figure 1) of the MEDLINE database of scientific articles on life science and biomedicine (used by the PubMed search engine,

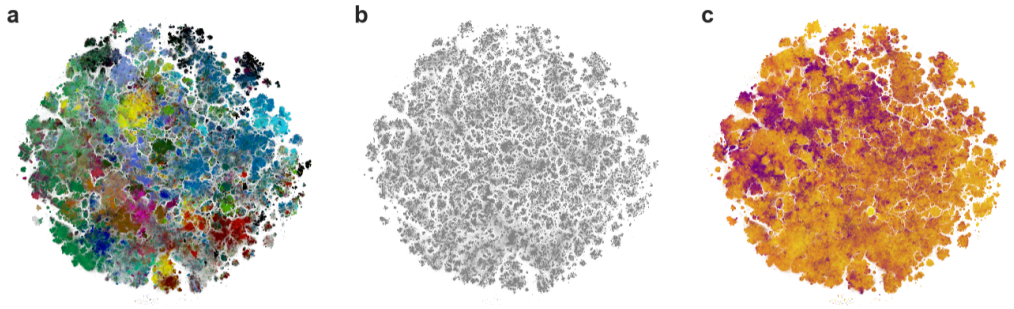

Figure 1: Our t-SNE embedding of the MEDLINE dataset ($n = 19$ M). Paper abstracts were transformed into TF-IDF representation, dimensionality was reduced to 300 with SVD and then to two with t-SNE (using uniform affinities on the $k = 10$ kNN graph, early exaggeration annealing, and 10,250 iterations). **(a)** Coloured using labels based on journal titles. **(b)** Uncoloured. **(c)** Coloured by publication year (dark: 1970 and earlier; light: 2021). See Figures A1–A3 for larger versions.

pubmed.ncbi.nlm.nih.gov). We benchmark different text preprocessing methods, embedding algorithms (t-SNE and UMAP) and t-SNE parameters, and show how to overcome challenges arising from the extremely large sample size ($n = 19$ M papers).

2 METHODS

2.1 t-SNE ALGORITHM, AFFINITIES, AND EXAGGERATION

The t-SNE objective is to find a low-dimensional representation $\mathbf{y}_i \in \mathbb{R}^2$ of high-dimensional points $\mathbf{x}_i \in \mathbb{R}^d$ that would preserve pairwise similarities (affinities) p_{ij} between points. The affinities are normalized to sum to 1 and are defined as follows:

$$p_{ij} = \frac{v_{ij}}{n}, \;\; v_{ij} = \frac{p_{i|j} + p_{j|i}}{2}, \;\; p_{j|i} = \frac{v_{j|i}}{\sum_{k \neq i} v_{k|i}}, \;\; v_{j|i} = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}\right). \tag{1}$$

The variance σ_i^2 of the Gaussian similarity kernel is chosen to yield a pre-specified value of the perplexity $\mathcal{P} = 2^{\mathcal{H}}$ of $p_{j|i}$ values, where $\mathcal{H} = -\sum_{j \neq i} p_{j|i} \log_2 p_{j|i}$ is the entropy. The affinities decay exponentially with distance, taking infinitesimal values for the majority of pairs of points. Modern t-SNE implementations (van Der Maaten, 2014; Linderman et al., 2019) explicitly truncate the affinities and use the k-nearest-neighbor (kNN) graph of the data with $k = 3\mathcal{P}$ as the input.

The affinity matrix can be further simplified by replacing Gaussian similarity kernel with the kNN graph adjacency matrix, by defining $v_{j|i} = 1$ if point j is within k nearest neighbors of point i and 0 otherwise. These uniform affinities with $k = \mathcal{P}/3$ typically yield a layout very similar to Gaussian affinities with perplexity $\mathcal{P}$ (Böhm et al., 2020), but require kNN graph with 9x smaller value of k.

In the low-dimensional space, affinities q_{ij} are defined using the Cauchy similarity kernel:

$$q_{ij} = \frac{w_{ij}}{Z}, \;\; w_{ij} = \frac{1}{1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2}, \;\; Z = \sum_{k \neq l} w_{kl}, \tag{2}$$

and the loss function is the Kullback-Leibler divergence $\sum_{ij} p_{ij} \log(p_{ij}/q_{ij})$. This loss is optimized using gradient descent, which makes close neighbors feel attractive forces, while other points feel repulsive forces. One standard optimization trick is to increase the strength of all attractive forces by a factor $\rho = 12$ in the first 250 iterations. This 'early exaggeration' (van der Maaten & Hinton, 2008) helps avoiding cluster fragmentation. Recent work (Kobak & Berens, 2019; Böhm et al., 2020) showed that in some cases, particularly for large datasets, it may be beneficial to keep some exaggeration on until the end of optimization. While high exaggeration helps preserving continuous manifold structures and emphasizes large clusters (Böhm et al., 2020), it comes with a price of distorting local neighborhoods.

2.2 MEDLINE DATASET AND EXPERIMENTAL SETUP

We downloaded the full MEDLINE database (∼250 GB) as XML files using the bulk download service (www.nlm.nih.gov/databases/download/pubmed_medline.html) on 26.01.2021. We used the Python `xml` package to extract PubMed ID, title, abstract, language, journal name, and publication date of all ∼30 M papers. We filtered out all 4.7 M non-English papers and all 7.6 M English papers with abstracts shorter than 250 or longer than 4000 symbols, leaving 19,016,308 papers for further analysis.

We labeled the dataset by selecting 21 frequent terms among journal names: bioinformatics, biology, cancer, cardiology, cell, chemistry, ecology, engineering, environmental, food, genetics, immunology, material, microbiology, neuroscience, nutrition, pediatric, psychiatry, psychology, surgery, and veterinary. Papers were assigned a label if their journal title contained that term (journal titles containing more than one term were assigned randomly to one of them), and left unlabeled otherwise. This resulted in 5,147,755 labeled papers (27%).

All experiments were performed in Python. We used Scikit-learn 0.24.1 (Pedregosa et al., 2011) for TF-IDF vectorizer, count vectorizer, and for truncated SVD. We used openTSNE 0.6.0 (Poličar et al., 2019), a Python reimplementation of FIt-SNE (Linderman et al., 2019), and UMAP 0.5.1 (McInnes et al., 2018). We used default parameters, unless specified.

All experiments were run on a server with 384 GB of RAM and 64 double-threaded 2.90 GHz CPU cores. Parsing the XML files took ∼10 h. Computing TF-IDF representation took 1 h. Truncated SVD took 4 h, t-SNE affinity calculation took 2 h, and subsequent t-SNE optimization took 2 h for the default 750 iterations, 4 h for 2250 iterations, and over 3 days for 10,250 iterations.

The code is available at `https://github.com/berenslab/pubmed-tsne-iclr`.

3 RESULTS

TF-IDF The abstracts were transformed into the bag-of-words representation C_{ij}, with elements denoting the number of times word j appears in abstract i. This was a sparse matrix of 19,016,308 $\times$ 4,114,381 size (0.003% non-zero elements). We tried various normalization approaches and used kNN classification accuracy to assess the quality of each representation (Table 1). We obtained the highest kNN accuracy (0.71) with the TF-IDF (term frequency – inverse document frequency) representation (Jones, 1972) with log-scaling, as defined in the scikit-learn implementation:

$$X_{ij} = (1 + \log_2 C_{ij}) \cdot \left(1 + \ln \frac{1+n}{1 + \sum_k (C_{kj} > 0)}\right) \text{ if } C_{ij} > 0, \text{ else } X_{ij} = 0, \tag{3}$$

where n is the total number of documents. In scikit-learn, the TF term is substituted by word counts C_{ij} for computational simplicity and instead, X_{ij} is row-normalized such that each row has norm 1.

Table 1: kNN classification accuracies for different preprocessing methods (with $k = 10$, test set of 500 labeled papers, and training set consisting of all remaining labeled papers). Row (4) is a standard preprocessing approach in single-cell transcriptomics (Luecken & Theis, 2019). Scaling term 100 was chosen as an approximate average abstract length (the average was 106 words).

Representation	**Equation**	**kNN accuracy**
Word counts	C_{ij}	0.49
Log-transformed counts	$\log_2(1 + C_{ij})$	0.54
Term frequencies (TF)	$C_{ij} / \sum_k C_{ik}$	0.54
scRNA-seq approach	$\log_2(1 + TF \cdot 100)$	0.62
Schmidt (2018)	$\max(0, \log_2(TF \cdot 100))$	0.43
TF-IDF without log-scaling	C_{ij} instead of $1 + \log_2 C_{ij}$ in Eq. 3	0.63
TF-IDF with log-scaling	X_{ij}, Eq. 3	**0.71**

SVD It is computationally infeasible to use the entire X_{ij} matrix for t-SNE. We therefore used singular value decomposition (SVD) to reduce dimensionality down to 300 — the largest number we could use given our memory resources. We employed SVD instead of PCA to avoid centering the sparse matrix. We did not observe any difference in performance between the `randomized` and `arpack` SVD solvers and used the latter solver for all further processing (randomized solver was 25% faster but used up 22% more RAM). After SVD, the kNN accuracy decreased only slightly, to 0.66 (Figures 2b, A4). At the same time, kNN recall (fraction of preserved $k = 10$ nearest neighbors, measured on a random subset of 500 papers) (Lee & Verleysen, 2009) was only 0.05, meaning that SVD was unable to accurately preserve nearest neighbors in our dataset (Figure A4). See Table A2 for 10 nearest neighbors in the TF-IDF and SVD spaces for one exemplary paper.

t-SNE affinities We found it impossible to use default t-SNE affinities with our sample size due to memory constraints, so we used uniform affinities (as implemented in openTSNE) with $k = 10$. Using a $n = 2$ M subset of the data, we confirmed that the resulting layout was nearly identical to using default affinities with perplexity 30 (Figure A5). Default affinities required 2.7x more RAM and took 1.05x more time.

t-SNE exaggeration We experimented with different values of exaggeration, $\rho \in \{1, 2, 4\}$, and found that $\rho > 1$ led to lower kNN recall (Figure 2a), in agreement with Böhm et al. (2020), and also lower kNN accuracy (Figure 2b). Visually, $\rho = 2$ led to somewhat increased cluster separation,

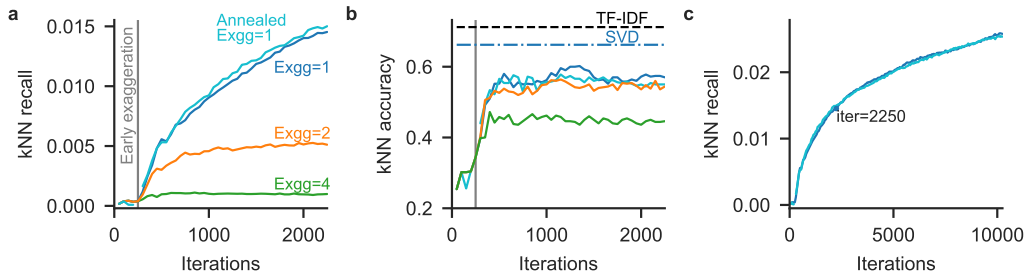

Figure 2: t-SNE embedding quality across iterations. **(a)** kNN recall compared to SVD ($k = 10$, test set size 10,000). **(b)** kNN accuracy for labeled papers ($k = 10$, test set size 10,000). **(c)** kNN recall during longer optimization.

whereas $\rho = 4$ lumped all points together (Figure A6), and neither provided clear benefits compared to $\rho = 1$. We conclude that exaggeration was not helpful for visualizing the dataset studied here. Without exaggeration, kNN accuracy after t-SNE was 0.57, while kNN recall was 0.014 compared to the SVD representation (Figure 2). We observed that kNN recall after t-SNE strongly decreased with increasing the sample size (Figure A7): from 0.2 at $n = 2000$ to 0.01 at $n = 19$ M.

UMAP UMAP (McInnes et al., 2018), a recent and very popular neighbor embedding method, yields very similar results to t-SNE with increased exaggeration, such as $\rho = 4$ (Böhm et al., 2020). Using a $n = 2$ M subset of the data, we confirmed that the UMAP embedding was qualitatively similar to the t-SNE embedding with $\rho = 4$ (Figure A8), and its kNN recall and kNN accuracy were both in between t-SNE values with $\rho = 2$ and $\rho = 4$, and worse than t-SNE without exaggeration (Figure A9). We used a subset of the data for this experiment because we were unable to run UMAP on the entire dataset due to memory constraints (UMAP uses Pynndescent for kNN graph construction and it needs more RAM than Annoy, employed by openTSNE).

Number of gradient descent iterations By default, gradient descent in openTSNE runs for 750 iterations (with learning rate $n/12$, Belkina et al., 2019). We observed that while kNN accuracy plateaued after around 300 iterations (Figure 2b), kNN recall did not plateau within 750 iterations (Figure 2a), so we used 2250 iterations for most experiments. Running optimization even longer increased kNN recall further (to 0.03 with respect to SVD, 0.01 with respect to TF-IDF, both after 10,250 iterations, Figure 2c), but took a long time: 4 hours for 2250 but over 3 days for 10,250 iterations. Note that the run time does not scale linearly because FIt-SNE approximation slows down when the embedding increases in size (Linderman et al., 2019). We felt that 2250 iterations provided a reasonable trade-off between accuracy and runtime.

Early exaggeration annealing By default, the early exaggeration factor is switched off after 250 iterations. We hypothesized that this abrupt transition may potentially 'trap' some points in sub-optimal locations. If so, gradually annealing early exaggeration factor down to 1 during iterations 125–250 could potentially improve the final embedding.

Using annealing, we observed almost no improvement in kNN recall (Figure 2a,c), however this metric may not be very sensitive to a minority of papers or clusters located suboptimally. We visually investigated the effect of annealing and found that while overall the embeddings with and without annealing looked very similar, there were some subtle changes on a smaller scale. One such example is shown in Figure 3. Without annealing (Figure 3a), there was an isolated island with predominantly blue papers (labeled 'materials') that ceased to exist when the exaggeration was annealed. Instead, its points got fused with two bigger regions of blue points (Figure 3b,c), suggesting that annealing helped them to drift to more meaningful locations. Two further examples of this effect can be seen in Figures A10 and A11.

For our final pipeline (Figure 1), we combined long optimization (10,250 iterations) with the early exaggeration annealing.

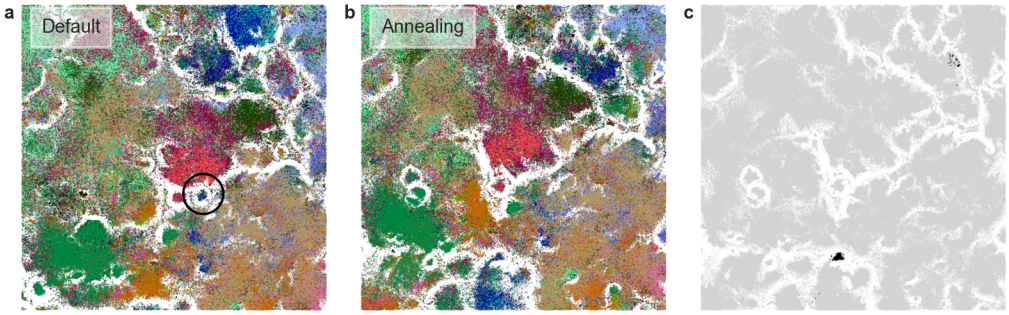

Figure 3: Annealing early exaggeration. **(a)** Isolated island (labeled 'materials') in the default embedding. **(b)** No such island after early exaggeration annealing. **(c)** Papers belonging to the original island highlighted after annealing.

4 Discussion

Here we developed a pipeline for making 2D visualizations of large document libraries, using the MEDLINE database of the entire biomedical literature as a use case. The pipeline is based on TF-IDF representation, SVD preprocessing, and t-SNE with uniform affinities, early exaggeration annealing, and extended optimization. Our final embedding (Figures 1, A1–A3) shows meaningful structure with good topic separation.

The closest neighbors were strongly distorted compared to the TF-IDF representation (Table A2), suggesting that such an embedding may not be suitable for literature queries. However, bigger-scale structure was meaningful, emphasizing small clusters containing 100s–1000s of papers on clearly defined topics.

In ongoing work, we found that random projection (Schmidt, 2018) outperformed SVD in terms of kNN recall, but resulted in poor t-SNE embeddings; future work is needed to explain this observation. It will also be interesting to replace TF-IDF with a pretrained language model such as PubMed BERT (Gu et al., 2021).

Acknowledgments

We thank Sebastian Damrich and Jan Niklas Böhm for comments and suggestions. This research was funded by the Deutsche Forschungsgemeinschaft (KO6282/2-1; through a Heisenberg Professorship, BE5601/4-1; and through the Excellence Cluster 2064 "Machine Learning: New Perspectives for Science", 390727645), and by the German Ministry of Education and Research (01IS18039A). The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Rita González Márquez.

References

Aleksandr Artemenkov and Maxim Panov. NCVis: noise contrastive approach for scalable visualization. In *Proceedings of The Web Conference 2020*, pp. 2941–2947, 2020.

Anna C Belkina, Christopher O Ciccolella, Rina Anno, Richard Halpert, Josef Spidlen, and Jennifer E Snyder-Cappione. Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature Communications*, 10(1): 1–12, 2019.

Jan Niklas Böhm, Philipp Berens, and Dmitry Kobak. Attraction-repulsion spectrum in neighbor embeddings. *arXiv preprint arXiv:2007.08902*, 2020.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-specific language model pretraining for biomedical

natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1): 1–23, 2021.

Geoffrey Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Neural Information Processing Conference*, volume 15, pp. 833–840. Citeseer, 2002.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 1972.

Dmitry Kobak and Philipp Berens. The art of using t-SNE for single-cell transcriptomics. *Nature Communications*, 10(1):1–14, 2019.

Dmitry Kobak, George Linderman, Stefan Steinerberger, Yuval Kluger, and Philipp Berens. Heavy-tailed kernels reveal a finer cluster structure in t-SNE visualisations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 124–139. Springer, 2019.

John A. Lee and Michel Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72(7):1431–1443, 2009.

George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. *Nature Methods*, 16(3):243–245, 2019.

Malte D Luecken and Fabian J Theis. Current best practices in single-cell RNA-seq analysis: a tutorial. *Molecular Systems Biology*, 15(6):e8746, 2019.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Maximilian Noichl. Modeling the structure of recent philosophy. *Synthese*, 198(6):5089–5100, 2021.

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, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pavlin G Poličar, Martin Stražar, and Blaž Zupan. openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding. *BioRxiv*, pp. 731877, 2019.

Benjamin Schmidt. Stable random projection: Lightweight, general-purpose dimensionality reduction for digitized libraries. *Journal of Cultural Analytics*, 2018.

Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th international conference on World Wide Web*, pp. 287–297, 2016.

Laurens van Der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

A Appendix

A.1 Supplementary Tables

Table A1: Nearest neighbor titles for Kobak & Berens (2019) (title: "The art of using t-SNE for single-cell transcriptomics") in the TF-IDF space ($d = 4$ M), SVD space ($d = 300$), and t-SNE space ($d = 2$). The t-SNE representation is the same as shown in Figure 1.

TF-IDF	Neighbor 1:	Application of t-SNE to human genetic data.
	Neighbor 2:	Heavy-tailed kernels reveal a finer cluster structure in t-SNE visualisations.
	Neighbor 3:	Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data.
	Neighbor 4:	Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets.
	Neighbor 5:	Performance comparison of dimensionality reduction methods on RNA-Seq data from the GTEx project.
	Neighbor 6:	Visualizing the spatial gene expression organization in the brain through non-linear similarity embeddings.
	Neighbor 7:	Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types.
	Neighbor 8:	Generalizable and Scalable Visualization of Single-Cell Data Using Neural Networks.
	Neighbor 9:	Visualization of Single Cell RNA-Seq Data Using t-SNE in R.
	Neighbor 10:	Current Projection Methods-Induced Biases at Subgroup Detection for Machine-Learning Based Data-Analysis of Biomedical Data.
SVD	Neighbor 1:	PyBDA: a command line tool for automated analysis of big biological data sets.
	Neighbor 2:	Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data.
	Neighbor 3:	scGEAToolbox: a Matlab toolbox for single-cell RNA sequencing data analysis.
	Neighbor 4:	Protocol for Identification and Removal of Doublets with DoubletDecon.
	Neighbor 5:	GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory networks.
	Neighbor 6:	Improved dropClust R package with integrative analysis support for scRNA-seq data.
	Neighbor 7:	Scrublet: Computational Identification of Cell Doublets in Single-Cell Transcriptomic Data.
	Neighbor 8:	Falco: a quick and flexible single-cell RNA-seq processing framework on the cloud.
	Neighbor 9:	zUMIs - A fast and flexible pipeline to process RNA sequencing data with UMIs.
	Neighbor 10:	scGMAI: a Gaussian mixture model for clustering single-cell RNA-Seq data based on deep autoencoder.
t-SNE	Neighbor 1:	Customized workflow development and data modularization concepts for RNA-Sequencing and metatranscriptome experiments.
	Neighbor 2:	RNA-Rocket: an RNA-Seq analysis resource for infectious disease research.
	Neighbor 3:	Deep Cap Analysis of Gene Expression (CAGE): Genome-Wide Identification of Promoters, Quantification of Their Activity, and Transcriptional Network Inference.
	Neighbor 4:	Workflow Development for the Functional Characterization of ncRNAs.
	Neighbor 5:	Exploring transcriptional switches from pairwise, temporal and population RNA-Seq data using deepTS.
	Neighbor 6:	Defining data-driven primary transcript annotations with primaryTranscriptAnnotation in R.
	Neighbor 7:	TRUFA: A User-Friendly Web Server for de novo RNA-seq Analysis Using Cluster Computing.
	Neighbor 8:	S-MART, a software toolbox to aid RNA-Seq data analysis.
	Neighbor 9:	A Guide to the Chloroplast Transcriptome Analysis Using RNA-Seq.
	Neighbor 10:	RNA-Seq in the Collaborative Cross.

A.2 SUPPLEMENTARY FIGURES

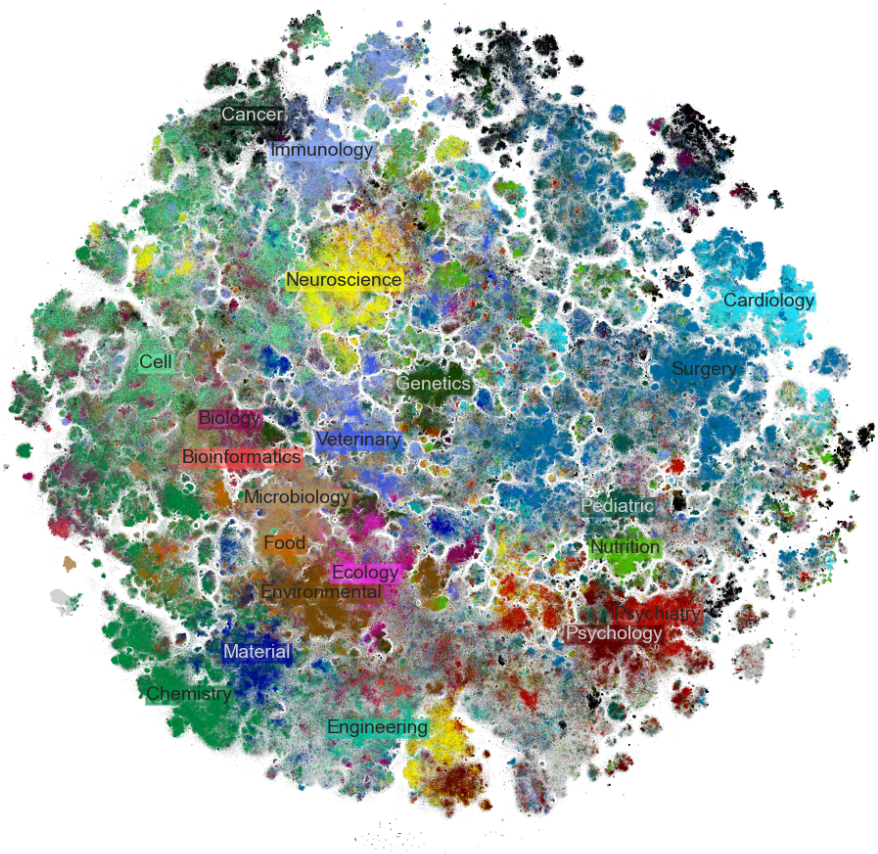

Figure A1: Our t-SNE embedding of the MEDLINE dataset ($n = 19$ M) coloured using labels based on journal titles. Unlabeled papers are shown in grey but are plotted underneath the labeled papers. For the visualization purposes, the labels are positioned in the location of the highest density of the corresponding class (obtained using a kernel density estimate).

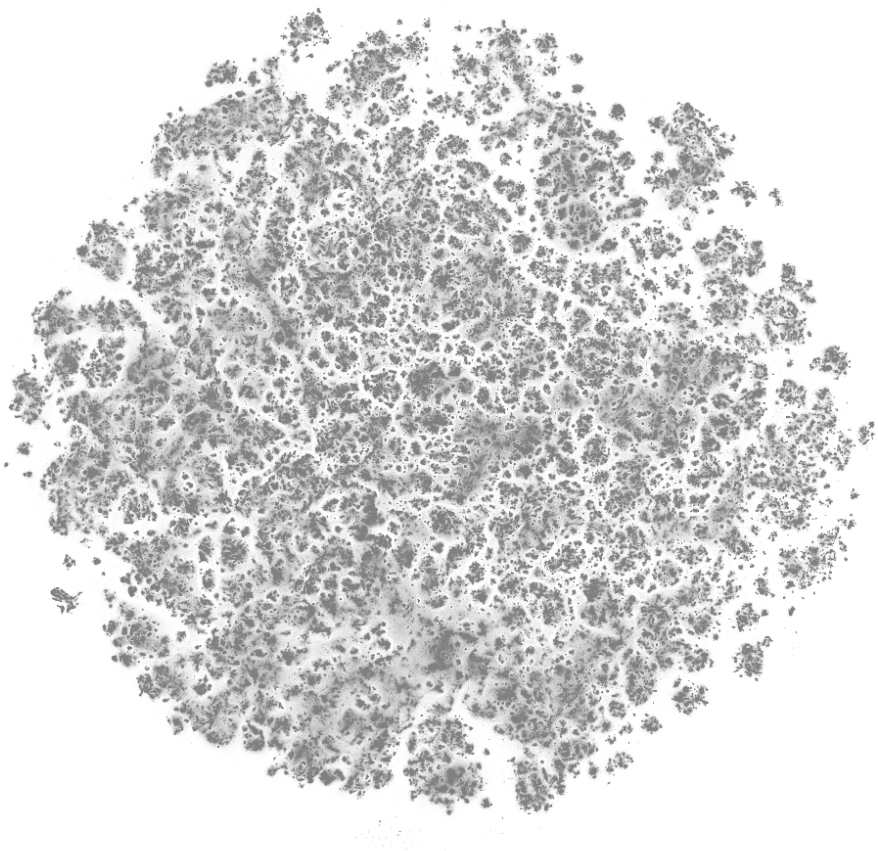

Figure A2: Our t-SNE embedding of the MEDLINE dataset ($n = 19$ M), without colours and with increased transparency of each dot, highlighting large number of small clusters.

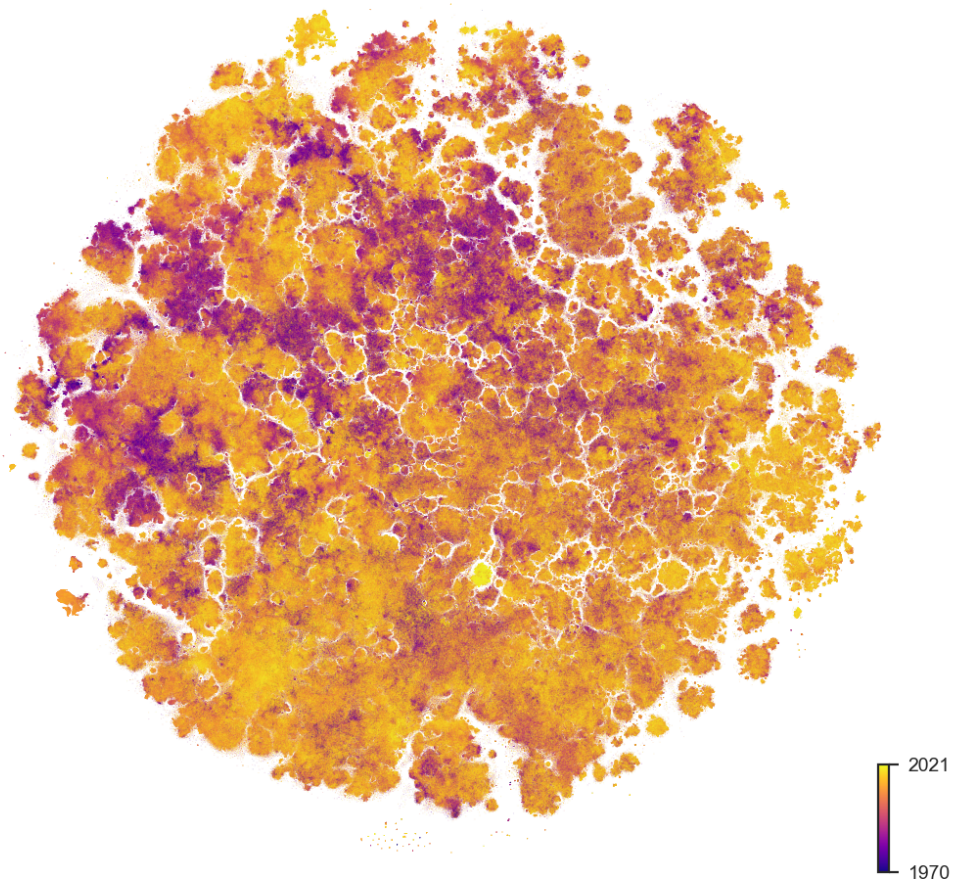

Figure A3: Our t-SNE embedding of the MEDLINE dataset ($n = 19$ M) coloured by publication year. The bright yellow island in the middle corresponds to papers on Covid-19.

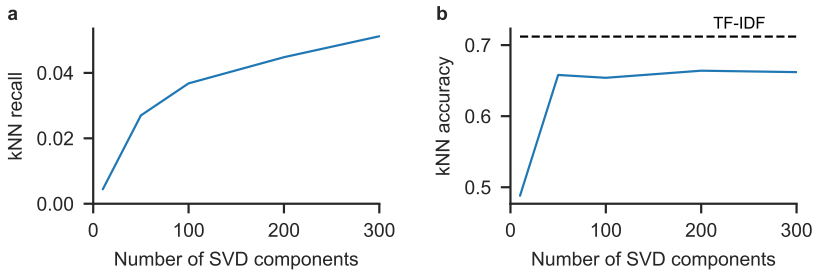

Figure A4: Representation quality after SVD. **(a)** kNN recall compared to TF-IDF ($k = 10$, test set size 500). **(b)** kNN accuracy for labeled papers ($k = 10$, test set size 500).

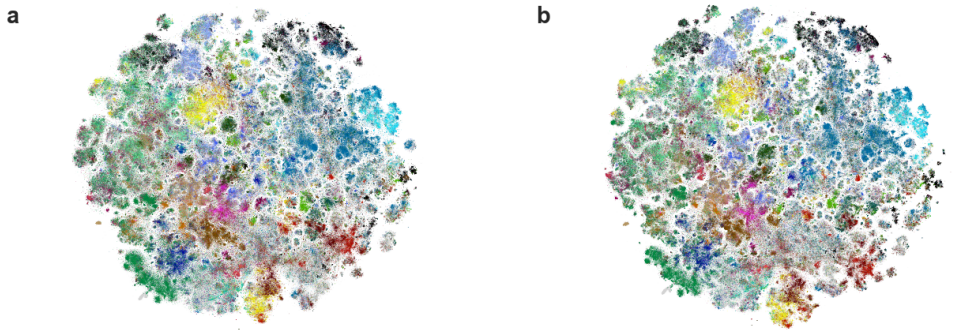

Figure A5: t-SNE embedding of a subset of the MEDLINE dataset ($n = 2$ M) using **(a)** the default affinities (perplexity 30), and **(b)** uniform affinities ($k = 10$), both with $\rho = 1$ and 2250 iterations. Runtimes and memory requirements: 65 min and 37 GB for the default affinities; 62 min and 14 GB for the uniform affinities.

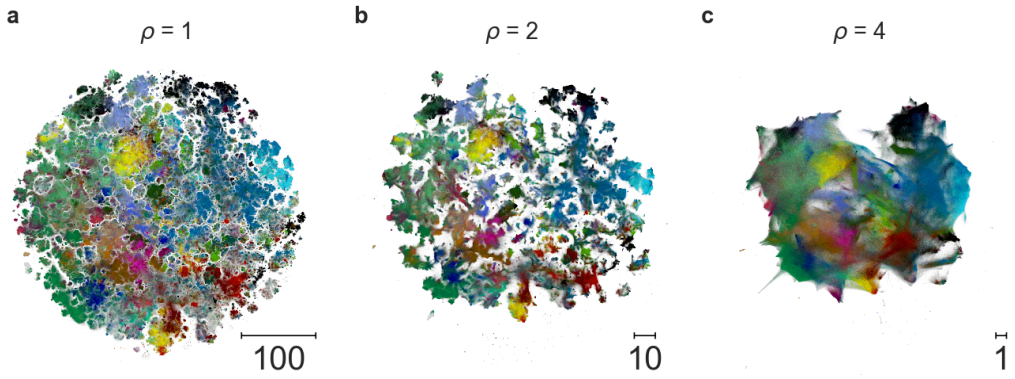

Figure A6: t-SNE embedding of the MEDLINE dataset ($n = 19$ M) for different values of exaggeration: **(a)** $\rho = 1$ (default), **(b)** $\rho = 2$, and **(c)** $\rho = 4$.

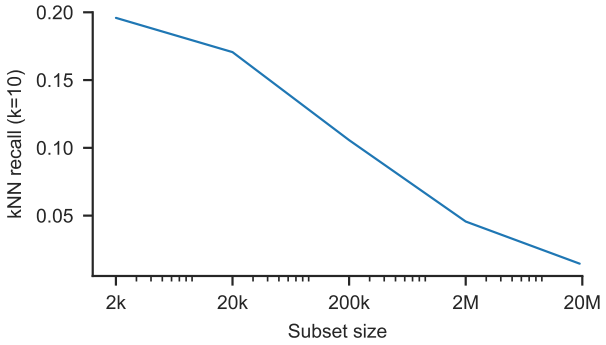

Figure A7: kNN recall ($k = 10$) after t-SNE for different subset sizes of the MEDLINE dataset.

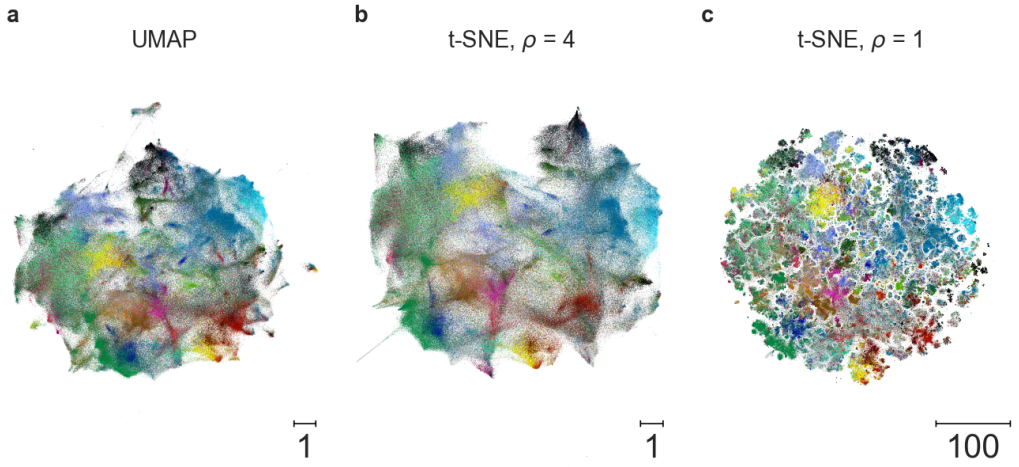

Figure A8: Embeddings of a $n = 2$ M subset of the MEDLINE dataset with **(a)** UMAP ($k = 10$), **(b)** t-SNE with $\rho = 4$, and **(c)** t-SNE without exaggeration ($\rho = 1$). The UMAP embedding was rotated to better align it with the t-SNE embeddings.

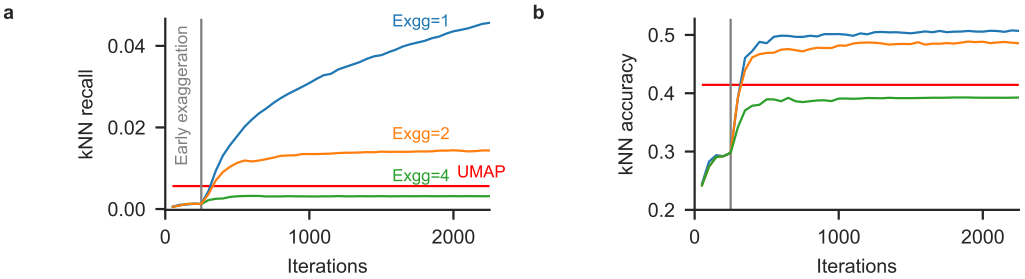

Figure A9: t-SNE and UMAP embedding quality across iterations for the $n = 2$ M subset. **(a)** kNN recall compared to SVD ($k = 10$, test set size 10,000). **(b)** kNN accuracy for labeled papers ($k = 10$, test set size 10,000).

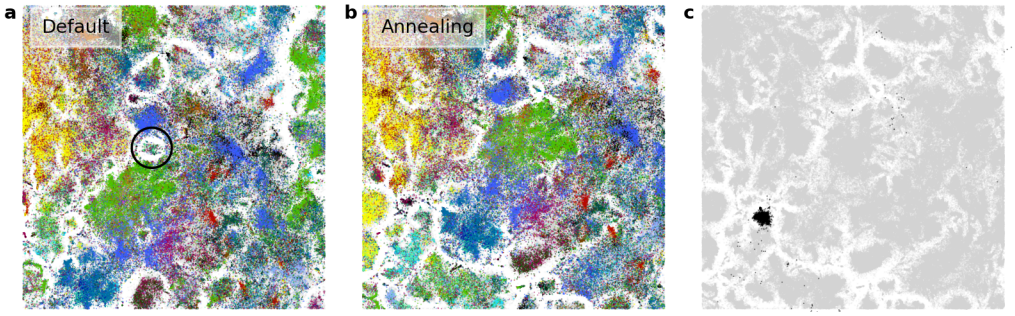

Figure A10: Annealing early exaggeration. **(a)** Isolated island in the default embedding. **(b)** No such island after early exaggeration annealing. **(c)** Papers belonging to the original island highlighted in black after annealing.

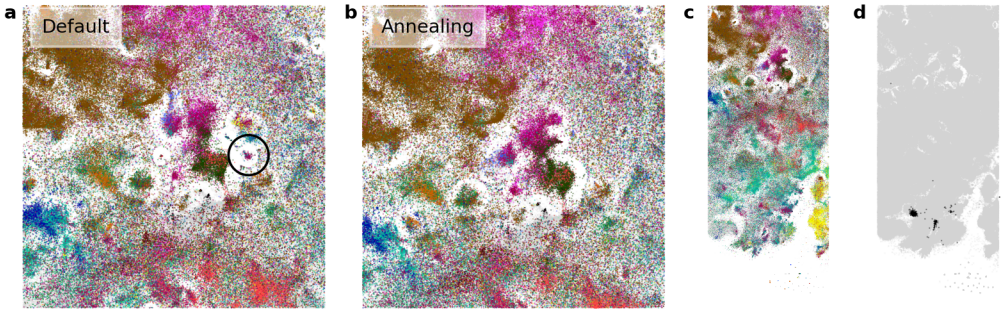

Figure A11: Annealing early exaggeration. **(a)** Isolated island in the default embedding. **(b)** No such island after early exaggeration annealing. **(c)** Zoom out of the embedding after annealing. **(d)** Papers belonging to the original island highlighted in black.