ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Improving t-SNE for applications on word embedding data in text mining

Bachelor Thesis Econometrie & Operationele Research

Supervisor: prof. dr. P.J.F. Groenen

Second Assessor: A. Castelein MSc

Christopher Claassen
456177cc@student.eur.nl

July 22, 2020

Abstract

There is an abundance of textual data in the present world. Recently proposed word embedding methods are able to provide meaningful quantitative word representations of textual data that can be used in various text mining tasks. However, this type of data is generally high-dimensional, such that it cannot be intuitively understood by looking at a graph. An interpretable graph can be made by using a technique called *t-distributed stochastic neighbour embedding* (t-SNE). In particular, this paper applies t-SNE to a word embedding generated by the word2vec model. Two extensions for t-SNE are proposed in this paper: local perplexities based on word similarities and a method for inserting a new point into an existing t-SNE. It is shown that these extensions work well and improve the usability of t-SNE for application in text mining.

Contents

1 Introduction

Obtaining insight from data is a fundamental part of scientific research. Before conducting any research, it is often a good first step to make a graph of two phenomena in order to detect a pattern between them. However, it is not feasible to do this for multiple high dimensional variables. An inherent problem of high dimensional data is that it cannot be visualized in a simple graph. Various dimension reduction methods have been used to address this problem. An example of this is *principal component analysis* (PCA; (Pearson, 1901; Hotelling, 1933)), which is a linear dimension reduction technique commonly used in multivariate analysis. PCA constructs a new orthogonal basis by performing an eigenvalue decomposition on the covariance matrix of the data. It then achieves a reduction in dimensions by discarding principal components that only represent a small portion of the variance in the original data. The visualization made by using the first two principal components can be represented in a graph, but the resulting points always retain some global structure of two dissimilar data points. This is not a useful property for visualization in a domain where one is mostly interested in local structure. Because of this, more sophisticated (non-)linear dimension reduction techniques have been developed.

In this research paper, I use a non-linear dimensionality reduction technique to visualize data resulting form a word embedding technique. This non-linear reduction technique is called *t-distributed stochastic neighbour embedding* (t-SNE) and was first introduced by Maaten and Hinton (2008). T-SNE is extension of *stochastic neighbour embedding* (SNE; Hinton and Roweis (2003)), which maps the original data to a low dimensional space by preserving the local structure of the data as much as possible. In particular, t-SNE improves SNE by solving the so called 'overcrowding' problem that causes data points to cluster in the middle of the graph. T-SNE and related methods can be seen as clustering methods because of the objective of local structure preservation.

T-SNE is applied on text representations generated by the word2vec model (Mikolov, Chen, Corrado, & Dean, 2013) in this paper. Word2vec is a word embedding algorithm that provides high dimensional representation of various words in order to process large amount of text efficiently. The algorithm primarily considers the local context a word appears in to produce the word embedding. The quality of the vectors will therefore largely depend on some local structure in the data. For this reason, it seems reasonable that t-SNE is able to visualize the word embedding data points in a good way, as t-SNE aims to retain the local structure of the data in favour of the global structure. Moreover, t-SNE is useful in this case, because researchers are often interested in finding clusters of words in a word embedding, as word embeddings do not label words automatically. Using t-SNE for word embeddings is straightforward, but it is not directly clear how one should can optimize t-SNE for this particular application. The core of this paper can therefore be best defined as the following research question:

How can t-SNE generated embeddings be improved for processing text via word embedding data?

The result of this paper is an improved t-SNE embedding algorithm for visualizing word representations. This paper finds that it is often useful to differentiate the perplexity parameter of t-SNE for individual points. This is done by combining the global perplexity value with a newly

constructed local perplexity value. Furthermore, this papers shows how a new word representation can be meaningfully placed in an existing t-SNE output, without running t-SNE on the entire dataset again. This is achieved by determining effective neighbours via word representation similarity weighing. The code used in this paper is written in the Julia language (Bezanson, Edelman, Karpinski, & Shah, 2017) and is available at this location. All figures and tables can be reproduced by individual functions that can be found inside the code. Appendix A contains instructions for the provided code.

The remainder of this paper consists of five sections. A brief overview of the related literature is provided in Section 2. Section 3 discusses the datasets that will be used in the research. Next, Section 4 describes the details of the t-SNE implementation, two extensions of t-SNE for text data and the evaluation criteria. This is followed by Section 5, where the replication and experimental results are discussed. Last, Section 6 concludes with an overview of this paper and recommendations for future works.

2 Related Works

This paper applies a non-linear dimensionality reduction technique on word embedding data. This is a popular topic, which means that various relevant papers are available. The most relevant papers will be presented in two subsections. Subsection 2.1 discusses the original implementation and limitations of the t-SNE technique and related methods. Subsection 2.2 contains an overview of the different architectures used in the word2vec word embedding algorithm.

2.1 T-SNE

Stochastic neighbour embedding (SNE; Hinton and Roweis (2003)) is a method for reducing the dimensionality of a dataset in a non-linear way. SNE allows for better visualization of data in comparison to standard principal component analysis (PCA; (Pearson, 1901; Hotelling, 1933)) if one is interested in representing the local structure of high-dimensional data well. As such, SNE can be seen as trying to capture the local structure of a high-dimensional data matrix X with a low-dimensional data representation Y. The resulting visualization can be seen as a clustering method because the individual points are positioned by being attracted to similar points and repelled by dissimilar points. This is achieved by modelling high-dimensional Euclidean distance by a probability p_{ij} and matching this with another probability q_{ij} from a low dimensional space. The initial positions in the low-dimensional space are originally randomly initialized.

The original SNE implementation uses non-symmetric probabilities, which is contrasted by symmetric SNE. For symmetric SNE, the p_{ij}'s from a high dimensional space are matched with the corresponding q_{ij}'s coming from a low-dimensional space through symmetric probabilities. The equations for calculating p_{ij} and q_{ij} between two datapoints for symmetric SNE are given by:

$$p_{ij} = \frac{\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)}{\sum_{k \neq l} \exp(-||\mathbf{x}_k - \mathbf{x}_l||^2/2\sigma^2)} \tag{2.1}$$

and

$$q_{ij} = \frac{\exp(-||\mathbf{y}_i - \mathbf{y}_j||^2)}{\sum_{k \neq l} \exp(-||\mathbf{y}_k - \mathbf{y}_l||^2)}, \tag{2.2}$$

where σ^2 is variance of the Gaussian curve, which is fixed at $\frac{1}{\sqrt{2}}$ for every q_{ij} as it less likely that the density will vary greatly between different points in that case.

The objective of symmetric SNE is to model every p_{ij} with a corresponding q_{ij} in a faithful manner. As stated previously, this faithful manner corresponds to preserving as much local structure as possible. In particular, symmetric SNE want to avoid modelling a small q_{ij} with a large p_{ij}, whereas this is less problematic the other way around. This feature is captured by the non-symmetric Kullback-Leibler divergence cost function that is given by:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{2.3}$$

The variance σ^2 of the Gaussian curve for every p_{ij} still needs to be specified. This can be done through the so called perplexity setting of symmetric SNE. The perplexity setting is a global setting that can be interpreted as a proxy for the effective amount of neighbouring points the algorithm needs to consider for every individual point. The variance of the Gaussian curve for p_{ij} can be found through fitting the user-specified perplexity via:

$$Perplexity(P_i) = 2^{P_i}, \tag{2.4}$$

where P_i is defined as the Shannon entropy that corresponds to:

$$H(\mathrm{P}_i) = -\sum_j p_{ij} \log_2 p_{ij}. \tag{2.5}$$

The objective of the optimization routine is to minimize the total cost of the Kullback-Leibler divergence for all points. This can be done by performing (stochastic) gradient descent of the corresponding gradient. The gradient and gradient update scheme, as defined in Maaten and Hinton (2008), are given by:

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j) \tag{2.6}$$

and

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)}). \tag{2.7}$$

where η indicates the learning rate and $\alpha(t)$ equals the current momentum at iteration t, which can be set as parameter initially.

However, the (symmetric) SNE method described above suffers from what is known as the 'crowding problem'. In short, this problem entails that SNE tends to clutter data points in the middle of the graph, because there is not enough low-dimensional space to model all the pairwise high-dimensional relations. *T-distributed stochastic neighbour embedding* (t-SNE; Maaten and Hinton (2008)) is a very similar non-linear dimension reduction technique that addresses the overcrowding problem of SNE. SNE and t-SNE work similar as they both map points from a

high dimensional space to a low dimensional space via a stochastic approach. However, t-SNE uses a Student t-distribution with a single degree of freedom, instead of the normal distribution of SNE, to model the q_{ij}'s. This adjustment addresses the crowding problem as the t-distribution has fatter tails. This means that t-SNE is able to better model large pairwise high-dimensional distances by large low-dimensional distances. The equation of q_{ij} is now given by:

$$q_{ij} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1}}. \tag{2.8}$$

This adjustment also changes the form of the gradient function, which is now easier to optimize. As it is strongly related, the updating scheme in equation (2.7) can still be used. The corresponding gradient of t-SNE is adjusted to:

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}. \tag{2.9}$$

A potential drawback of t-SNE is that the objective function is not convex. It can therefore occur that a qualitatively lacking embedding is generated due to the optimization routine getting stuck in poor local minima. This concern can partially be addressed by putting certain restrictions on the learning rate and momentum. Furthermore, Maaten and Hinton (2008) show that exaggerating the value of early p_{ij}'s can also be helpful. Other solutions to address this problem will be discussed later.

A second drawback of T-SNE is the total running time of $\mathcal{O}(n^2)$. T-SNE works well for visualizing smaller datasets, but it is not a feasible method for visualizing larger datasets that are found in practical research. For this reason, various theoretical (Van Der Maaten, 2014; Linderman, Rachh, Hoskins, Steinerberger, & Kluger, 2017) and hardware based (Pezzotti et al., 2018; Chan, Rao, Huang, & Canny, 2019) improvements have been proposed. These methods all reduce the running time of t-SNE significantly such that it can still be practically used on large datasets.

2.2 Word embeddings

Word embeddings are a collection of methods that map a large amount of words to a single vector space. This results in a vector representation for a word that contains, for example, 300 features for every word. This has the advantage that further mathematical manipulation can be done through vector arithmetic. Moreover, every vector retains some semantic and/or syntactic relations to other words. For example, $\text{vec}(Germany) - \text{vec}(Berlin)$ results in a vector that is remarkably similar to the result of $\text{vec}(Austria) - \text{vec}(Vienna)$. Word embeddings outperform traditional co-occurrence counting methods for various text mining tasks, as word embeddings are able to leverage more information. Word embeddings can be used in various practical applications in text mining, like building a recommendation system based on words an individual uses or classifying the sentiment of a review based on the words it contains.

Word2vec is a widely used machine learning algorithm for building word embedding models. The core architectures of the algorithm, called CBOW and Skip-gram, were first introduced by Mikolov, Chen, et al. (2013). The algorithm does not use a restricted Boltzmann machine to

reconstruct data, but rather attempts to predict the word given its local context or vice versa. As such, word2vec is able to make word embeddings by either predicting a word given its context (CBOW) or the context given the word (Skip-gram). Both CBOW and Skip-gram learn their word embeddings through a shallow, two-layer neural network. This neural network is trained on text datasets that are known as the training corpus. The related loss functions are given by:

$$\text{CBOW} : L = \frac{1}{v}\sum_{t=1}^{V} \log\, p(w_t|w_{t-c},\ldots,w_{t-1},w_{t+1},\ldots,w_{t+c}) \tag{2.10}$$

and

$$\text{Skip-gram} : L = \frac{1}{v}\sum_{t=1}^{V}\sum_{i=t-c,i\neq t}^{t+c} \log\, p(w_i|w_t) \tag{2.11}$$

where V is the size of the learned vocabulary, $[-c,c]$ represents the context window of the target word w_t and $p(w_{t+j}|w_t)$ is softmax function defined by:

$$p(w_O|w_I) = \frac{\exp(v'_{w_O} v_{w_I})}{\sum_{w=1}^{W}\exp(v'_{w_O} v_{w_I})}, \tag{2.12}$$

where w_O is the word representation in the output layer and w_i is the word representation in the input layer.

The softmax function is impractical in the sense that it requires too much computing time for large training sets. For this reason, Mikolov, Sutskever, Chen, Corrado, and Dean (2013) developed an alternative method called *Skip-Gram with Negative Sampling* (SGNS). SGNS attempts to distinguish a target word w_O from a noise distribution $P_n(w)$. The SGNS objective function is given by:

$$\log\,\sigma(v'_{w_O} v_{w_I}) + \sum_{i=1}^{k} \mathbf{E}_{w_i\sim P_n(w)}[\log\,\sigma(v'_{w_O} v_{w_I})]. \tag{2.13}$$

The quality of the resulting word embedding vectors is usually assessed by calculating the cosine similarity between various words to see if the most similar ones match the excepted most similar words. The cosine similarity is used in favour of the Euclidean distance, because the word vector magnitude is correlated with the amount of times the word appears in the training corpus (Schakel & Wilson, 2015). Therefore, one is interested in whether certain words appear together often. The cosine similarity is usually normalized to lie within the interval $[0,1]$, where a cosine similarity of 0 or 1 indicates a very dissimilar or similar word respectively. The cosine similarity between word vectors $\mathbf{x}_i$ and $\mathbf{x}_i$ is defined as:

$$\cos(\theta) = \frac{\mathbf{x}'_i\mathbf{x}_j}{||\mathbf{x}_i||\;||\mathbf{x}_j||}. \tag{2.14}$$

There are other word embedding methods that mirror the performance of SGNS, such as GloVe (Pennington, Socher, & Manning, 2014) and fastText (Bojanowski, Grave, Joulin, & Mikolov, 2017). GloVe produces word embeddings by factorizing the global word co-occurrences matrix in a separate word and context matrix, whereas fastText considers a subword version of Skip-Gram to produce embeddings at the n-gram level. These word embedding methods

provide similar word representations in practice when given a sufficiently large training corpus for different tasks. There is no clear best method among these word embedding models. This paper will use data generated by a SGNS model, as it is the most widely studied and used in the scientific literature.

3 Data

Maaten and Hinton (2008) use the MNIST dataset for the evaluation, which is a standard dataset in the literature. It contains 60000 grayscale images of handwritten digits. For the sake of replication, it is enough to use a smaller dataset. This is done because it takes a lot of time to evaluate the entire dataset. I will therefore use the Iris flower dataset that was originally used in Fisher (1936). This dataset consists of four measurement of 150 different flowers. It is originally only possible to replicate the original results to a certain degree. This is because t-SNE produces slightly different results due to its inherent random initialization and the form of the cost function. This problem is partially addressed by fixing the initialization to the same values in both cases.

The proposed extensions will make use of different data produced by the word2vec word embedding method. These word embeddings can either be trained manually or retrieved from a general pre-trained dataset. A pre-trained dataset has the advantage that is based on a large training set, which means that it can use a lot information for learning word representations. On the other hand, a manually trained model is potentially better suited for a particular research problem at hand. As the aim of this research is to introduce general improvements for visualizing word embeddings models, a pre-trained word2vec model will be used. This is the Google-news-300 dataset (Google, 2013), which is trained on a corpus of roughly 100 billion words that contains 3 billion running words using the previously discussed SGNS method. The dataset contains about 3 million word representation vectors, where each word representation has a feature dimensionality of 300.

A typical requirement for the practical use of word embeddings is that the raw word representations are clustered in groups of related terms as these groups are generally not provided. A representative graph produced by t-SNE is ideal for this purpose. A selection of 601 word representations of the original dataset will be used to evaluate the proposed extensions. These words belong to one of eleven categories. Examples of these categories are *countries*, *food* and *university subjects*. These groups are not evenly balanced nor necessarily related to one another. This is intentionally done such that the effect of the extensions on groups with different characteristics can be examined. Figure 1 contains a scatter plot based of the first two principal components. It can be seen that the first two principal components do not provide a clear separation of clusters for all the different groups, especially when one does not consider the colours the labels provide. These labels are assigned manually and will not be used for the t-SNE algorithm. More details on the composition of the different word groups can be viewed in Appendix B. This subset of the Google news dataset can be generated and viewed at the same location as the project code.

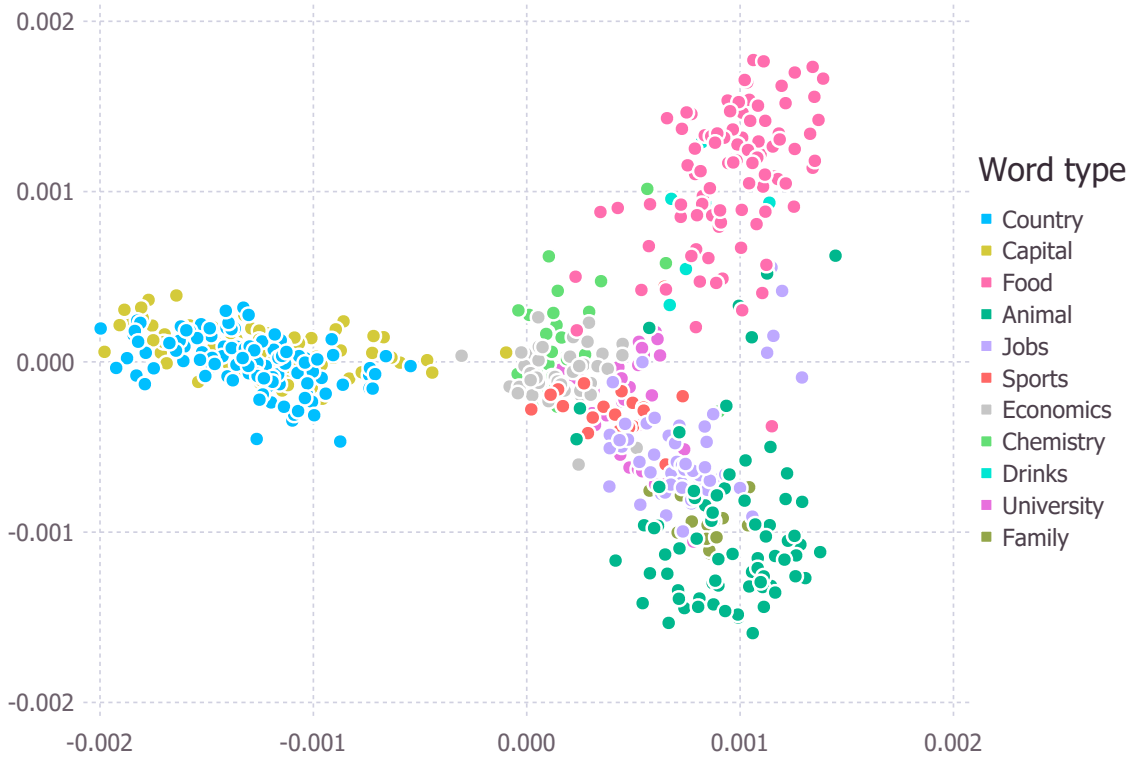

Figure 1. Scatter plot of the first two (scaled) principal components, coloured by word group

4 Methodology

This section contains a discussion of the methodology used in this paper. It is split in three subsections on implementation, extensions and evaluation. Subsection 4.1 discusses the original implementation and limitations of the t-SNE technique. Two new extensions are presented in Subsection 4.2. The first extension can be describes as local perplexity scaling and the second extension introduces a method for efficiently inserting new points. Last, Subsection 4.3 contains an overview of the evaluation criteria that are utilized to validate the resulting embeddings.

4.1 An improved t-SNE implementation

T-SNE is widely studied and used since its introduction. Because of this, the original t-SNE implementation of Maaten and Hinton (2008) is dated in the sense that there have been various improvements to the algorithm over the years. I will know discuss two such features that I have incorporated in my implementation of t-SNE. This implementation of t-SNE contains all to be discussed features and extensions, which are written in the Julia language (Bezanson et al., 2017) due to its high computing performance.

The original t-SNE implementation initializes the low-dimensional coordinates by performing random draws from a $N(0, 0.001^2)$ distribution. My implementation adds three additional initialization settings for the algorithm based on either PCA, classical MDS or a custom initialization. These initializations are scaled to be more comparable to the original random initialization. Using either of these first two initialization provides three advantages over a random initialization.

Firstly, they address the reproducibility problem caused by initial randomness. Secondly, they prevent a poor embedding that is caused by a bad random draw. Thirdly, they result in an embedding where a higher portion of the global structure is retained (Kobak & Berens, 2019).

The original cost function of t-SNE is non-convex which makes it prone to getting stuck in local minima during optimization. Maaten and Hinton (2008) originally address this problem by using early exaggeration in the gradient descend algorithm. This results in replacing the original p_{ij}'s with $\alpha\ p_{ij}$ ($\alpha > 1$) such that the embedding is improved. The original implementation uses $\alpha = 4$ for the first 50 iterations, but Linderman and Steinerberger (2019) show alternative early exaggeration settings that provide better results in a more general setting. Linderman et al. (2017) introduce the concept of late exaggeration, which applies the principle of early exaggeration to the last few optimization iterations. Late exaggeration potentially results in better defined clusters. My implementation features both early and late exaggeration options for the reasons listed above.

4.2 Extensions

4.2.1 Multi-scale local perplexities based on word similarities

The perplexity parameter of t-SNE can be interpreted as a measure of the number of effective neighbours the algorithm considers, which is a similar concept to the number of nearest neighbours used in other algorithms. The perplexity setting is a global parameter in the original implementation. However, a common feature of language is that individual words have a different amount of semantically or syntactically related words. This means, for example, that we might want the word *Greece* to consider all other countries for an embedding, whereas as we want the word *Twitter* to consider a quantitatively smaller amount of big social media platforms. Moreover, training a word embedding model for a specific task implicitly implies the existence of quantitatively unbalanced groups, as we want to capture as many terms related to the task at hand. As such, it is reasonable to assume that the quality of the embedding can be improved by implementing this information.

Following this, it is useful to tweak the perplexity setting of an individual word representation to a certain extent. This can be done by averaging the p_{ij} based on a global perplexity and a different p_{ij} based on a local perplexity. Kobak and Berens (2019) show this is approximately equal to replacing the Gaussian kernel $\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)$ of p_{ij} in t-SNE by a multi-scale kernel defined by Lee, Peluffo-Ordóñez, and Verleysen (2015) as:

$$\frac{1}{\sigma}\exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}) + \frac{1}{\tau}\exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\tau^2}), \tag{4.1}$$

where σ and τ are fitted such that they equal the global and local perplexity values respectively.

The global and local perplexity values can easily be set as an algorithmic parameter in this framework. However, it is not immediately clear how individual local perplexity values can be determined from the data. The goal of this extension is not to precisely determine all individual perplexities, but rather to differentiate the relative perplexities in such a way that it improves the visualization. In other words, the local perplexity should complement the global perplexity. This can be achieved by first assigning a perplexity score to each word based on the cosine

similarity surpassing some threshold θ_1, after normalization. More specifically, this means that the perplexity score of word representation i is determined by:

$$Perplexity\ score_i = \sum_{j \neq i} \mathrm{I}[\cos\_\mathrm{sim}(\mathbf{x}_i, \mathbf{x}_j) \geq \theta_1], \tag{4.2}$$

where $\mathbf{x}_j$ ($j \in \{1, \ldots, N\}$) are all n representation in the original data and θ_1 is a threshold value that obeys $0 \leq \theta_1 \leq 1$. A global value of θ_1 will likely not work well for all word representations in the dataset because the similarity vectors of different word embeddings are differently distributed. θ_1 should have a specific value for each word representation in order to account for this issue. θ_{1i} can accordingly be defined as:

$$\theta_{1i} = \mu_{\mathbf{s}_i} + k\sigma_{\mathbf{s}_i}, \tag{4.3}$$

where $\mu_{\mathbf{s}_i}$ and $\sigma_{\mathbf{s}_i}$ represent the mean and standard deviation of the similarity vector of word representation i respectively and k corresponds to a settable parameter. A higher value of k leads to lower perplexity scores. In practice, one can set k to be equal to either 2 or 3, while keeping in mind the resulting value of θ_{1i} should not surpass the maximum similarity value of 1.

The calculated perplexity scores can be used to construct individual local perplexities. This is done by adjusting the entire high dimensional similarity matrix. Similarity values of each column are set to 0 until only the most similar words embeddings are still represented with a positive similarity. More specifically, each column only retains a number of positive similarities that is equal to its perplexity score. Afterwards, the local perplexity matrix is normalized such that the columns each sum up to a value of 1. Consequently, the adjusted perplexity matrix is not constructed like in equation 4.1 by averaging perplexities of different Gaussian kernels. Instead, this adjusted perplexity matrix is constructed by averaging the fitted perplexities of the original Gaussian kernel based on the global perplexity setting and the matrix of local perplexities described in this paragraph.

4.2.2 Adding a new point to an existing t-SNE output

A practical problem of t-SNE is that a not already embedded point cannot directly be inserted into an existing embedding as t-SNE does not provide an explicit mapping. Nevertheless, we are often interested in looking at which already present terms are similar to a newly introduced term. For this application, we are not specifically interested in the global structure of this new term as we want to match the new term to specific terms. We would want to insert a point in such a way that it mostly depends on the high dimensional data of directly related terms. This is a similar objective as t-SNE. However, running t-SNE again on the expanded dataset is not an efficient way of addressing this problem because of the running time.

It is possible to insert points in t-SNE output in a different way by making use of the geometric median. The geometric median is the point that minimizes the total distance between itself and the other points in the embedding. The cost function of the geometric median for a new point $\mathbf{z}$ in an embedding with n points is given by:

$$C(\mathbf{z}) = \sum_{i=1}^{n} ||\mathbf{y}_i - \mathbf{z}||, \tag{4.4}$$

where this equation can be minimized by using the Weiszfeld algorithm (Aftab, Hartley, & Trumpf, 2014). This algorithm updates an initial solution $\mathbf{z}^0$ by using the update:

$$\mathbf{z}^{t+1} = \frac{\sum_{i=1}^{n} w_i^t \mathbf{y}_i}{\sum_{i=1}^{n} w_i^t} = \frac{\sum_{i=1}^{n} ||\mathbf{z}^t - \mathbf{y}_i||^{-1} \mathbf{y}_i}{\sum_{i=1}^{n} ||\mathbf{z}^t - \mathbf{y}_i||^{-1}}, \tag{4.5}$$

where $\mathbf{z}^0$ is the initial solution that can be set equal to the (weighted) mean and w_i^t denotes the vector of weights that is set equal to the inverse of the norm for the unweighted problem.

However, the resulting solution for $\mathbf{z}$ has two disadvantages in this case. The first disadvantage is that it does not provide a unique solution for multiple different points, whereas t-SNE does provide unique embeddings. The second disadvantage is that this solution considers every other point, whereas t-SNE effectively neglects a point if the high-dimensional pairwise distance is too big. These drawbacks can be addressed by adjusting the weights w_i^t and by only considering a set of effective neighbours, which we now turn to.

As previously mentioned, it makes sense to only consider some directly related words for the positioning of a new word from both a t-SNE and language point of view. For example, we prefer that the predicted position of the newly added word *Berlin* is close to the word *Germany*, whereas we do not want it to be close to the word *towel*. We can do this by limiting the number of considered points, or effective neighbours, in equation (4.4) to k, where it holds that $k \leq n$. This adjusts the cost equation (4.4) to:

$$C(\mathbf{z}) = \sum_{i \in E} ||\mathbf{y}_i - \mathbf{z}||, \tag{4.6}$$

where E denotes the set of effective neighbours, such that only k neighbours are relevant to the minimization of the cost function.

As we want to represent the high-dimensional data, it makes sense to only consider the t-SNE embeddings of the k most (cosine) similar words in the high-dimensional data. The value of k can either be set manually, or equal to the perplexity score of equation (4.2). This last option effectively allows us to add a point to the embedding by only considering the position of the words that have a cosine similarity of at least θ_{1i} in the high-dimensional data.

It is still possible that a solution based on the restricted geometric median is not unique, because it is possible that two word representation share the exact same set of effective neighbours in the high-dimensional space. This issue can be addressed by changing the weights in equation 4.5 such that the solution becomes a weighted geometric median. The weights can be rescaled by multiplying them by similarity s_i which results in:

$$s_i w_i^t = \frac{s_i}{||\mathbf{z}^t - \mathbf{y}_i||}, \tag{4.7}$$

where s_i represents the cosine similarity between word i and the newly added target word, after rescaling all s_i such that the largest similarity equals 1. This creates a unique solution by incorporating an additional bit of information of the high-dimensional data. However, this

does not affect the solution significantly due to fact that the similarities are positioned relatively close together on a linear scale. Moreover, they are all fairly large if they were picked according to being the most similar to the new word. Multiplying all similarities by a constant does not solve this problem as only the relative values of all similarities are important. I thus consider two non-linear functions of s_i that reduce the relative importance of lesser related terms, such that it becomes either a power weighing function $f_1(s_i, p)$:

$$f_1(s_i, p)w_i^t = s_i^p w_i^t = \frac{s_i^p}{||\mathbf{z}^t - \mathbf{y}_i||} \tag{4.8}$$

or an exponential weighing function $f_2(s_i, p)$:

$$f_2(s_i, p)w_i^t = \frac{p^{s_i} - 1}{p - 1} w_i^t = \frac{p^{s_i} - 1}{(p - 1)||\mathbf{z}^t - \mathbf{y}_i||}, \tag{4.9}$$

where p is a tuning parameter of which the effects are illustrated in Figure 2. This figure shows the shape of the weighing function compared to the base similarity input for various values of p. It can be seen in the figure that the power weights quickly push the similarities towards 0, whereas the exponential weights remain relatively closer to the base similarities for more extreme values of p. Both weighing function have some interesting properties for particular p: $p = 0$ gives uniform weights and $p = 1$ gives the original similarities. As such, it is also possible to acquire the unweighted or similarity weighted solutions through these non-linear functions.

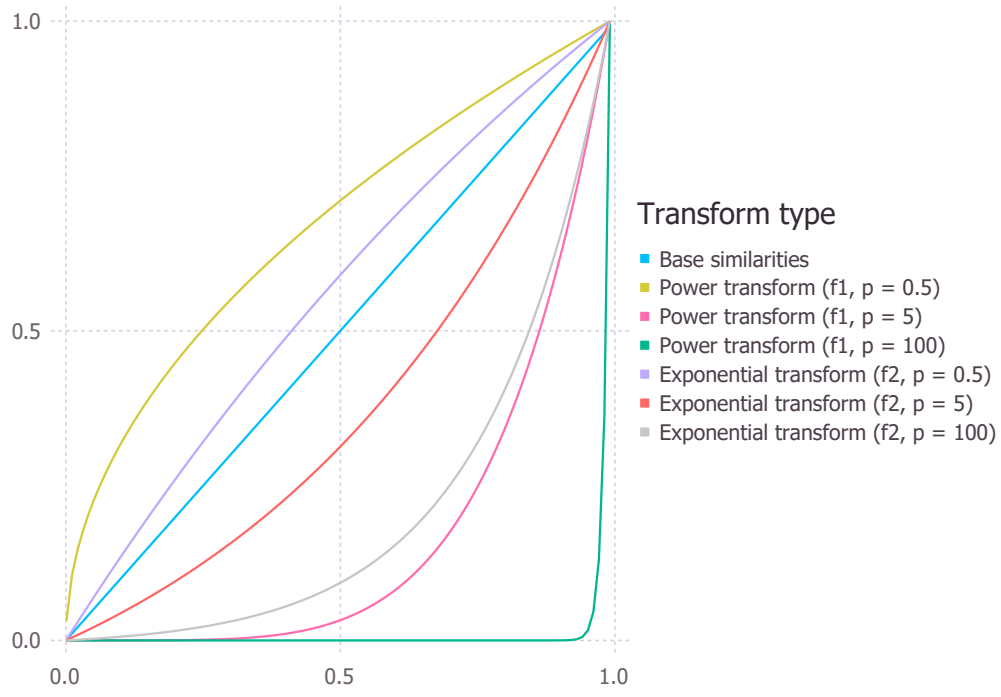

Figure 2. Example of the non-linearly rescaled similarities for various parametric values

The tuning parameter p in equation (4.8) can be interpreted by rewriting f_1 as:

$$p = \frac{\log(f_1(s_i, p))}{\log(s_i)} = \frac{\log(\theta_2)}{\log(s_{target}) - \log(s_{max})}, \tag{4.10}$$

where the interpretation is as follows: p sets all similarities according to equation (4.8) such that some target word is weighed exactly θ_2 times as effective as the most similar word in the embedding. This tells us, for example, how we can set p in such a way that the least similar effective neighbour is exactly 0.05 times as important as the most similar effective neighbour. This interpretation holds for p in equation (4.9) as well, but the value of p can only be acquired numerically in that case. It is important to note that the primary interest of weighing similarities is to acquire a useful relative difference between these similarities. Both equation (4.8) and equation (4.9) can be used for that purpose, where equation (4.8) is preferred if we want to discard most of the information of words with relatively low similarity. Lastly, the power weighting procedure can also be used as a substitute for the effective neighbour selection by setting the similarity score of some target word in the whole similarity set to near zero. This effectively discards the information of all words that are less similar than that target word in the high-dimensional space.

In conclusion, a new point can be added to an existing t-SNE embedding by using the effective neighbour similarity weighted geometric median, which has a cost function defined as:

$$C(\mathbf{z}) = \sum_{i \in E} f_k(s_i, p)||\mathbf{y}_i - \mathbf{z}||, \tag{4.11}$$

where $\mathbf{z}$ is a point that is added in an embedding consisting of all $\mathbf{x}_i$, E denotes the set of effective neighbours and $f_k(s_i, p)$ corresponds to a transformation function in either equation (4.8) or equation (4.9).

4.3 Evaluation criteria

Both extensions can be evaluated by looking at whether the (adjusted) t-SNE embedding is able to retain high-dimensional information in the low-dimensional embedding. This can be done informally by inspecting the graph, or formally, by making use of evaluation metrics. Of particular interest is the ability to retain local and global structure in an embedding generated by using local perplexities. In addition, the quality of a newly inserted point can also be assessed.

The quality of the local structure of a single point in the embedding can be evaluated by looking at the fraction of words that appear in the set of k closest points in both the low- and high-dimensional space. The chosen integer value k should not be too high in this case, as it is used to measure local structure. Both sets of k closest points can be obtained by using a k-nearest-neighbour (kNN) algorithm. The overall local structure evaluation metric μ_{local} can consequently be defined as the average of all these fractions for each word representation in the embedding:

$$\mu_{local} = \frac{1}{n} \sum_{i=1}^{n} F_i, \tag{4.12}$$

where F_i represents the fraction of words that appear in both sets of k closest points for word representation i. The quality of the local structure of a newly inserted point can be evaluated in a similar way by comparing the sets of k closest points for every newly added point individually.

The overall quality of the global structure of the embedding can be evaluated by measuring

to what extent the ordering of pairwise distances in the high-dimensional dataset is retained in the low-dimensional embedding. The quality of the global structure for an individual point i can be assessed by comparing the vectors $\mathbf{d}_L$ and $\mathbf{d}_H$, which contain that all pairwise distances between i and all other points in either the low or high dimensional space. The evaluation of $\mathbf{d}_L$ and $\mathbf{d}_H$ can be done by calculating the Spearman rank-order correlation, which is defined by:

$$\rho_i = \frac{\text{cov}(\text{rank}_{\mathbf{d}_L}, \text{rank}_{\mathbf{d}_H})}{\sigma_{\text{rank}_{\mathbf{d}_L}} \sigma_{rank_{\mathbf{d}_H}}}, \tag{4.13}$$

where $\text{rank}_{\mathbf{d}}$ represents the rank ordering vector of $\mathbf{d}$. The evaluation metric μ_{global}, which represents the overall quality of the global structure, can consequently be calculated as:

$$\mu_{global} = \frac{1}{n} \sum_{i=1}^{n} \rho_i. \tag{4.14}$$

The quality of the global structure of a newly inserted point can be evaluated in a similar way by calculating the Spearman rank-order correlation for every newly added point individually.

5 Results

The experimental results will be presented and discussed in this section. It consists of two subsections. Subsection 5.1 discusses the faithfulness of the t-SNE implementation that is used in this paper. The experimental results and corresponding graphs are presented in Subsection 5.2. All results are obtained using Julia version 1.4.1 on a system with an Intel Core i5-4690K CPU and 16GB of RAM.

5.1 Replication

It is useful to consider to what extent the output of this paper's t-SNE implementation resembles the output of the by van der Maaten recommended Julia t-SNE implementation. Figure 3 shows the output of running t-SNE on the Iris flower dataset by using a PCA initialization and a perplexity value of 15 for both implementations. It appears that the graphs show the same three clusters, but the graphs can certainly not be called identical. However, the average local structure retained is almost the same, as can be seen in Table 1. The difference in graphical representation is mainly caused by the inclusion of late exaggeration in the algorithm of the right figure. Disabling this feature would make the graphs a lot more similar, but this feature is useful for processing textual data. This is because late exaggeration causes the final embedding to show more well-defined clusters, as can be clearly seen in Figure 3. The benchmark implementation takes significantly less time to produce the t-SNE output. This is primarily due to the fact that the benchmark implementation uses various features of Julia to accelerate costly calculations. Another interesting result of Table 1 is the fact the first two principal components are also able to retain a relatively large amount of local structure. This is primarily caused by the fact that the difference between low- and high-dimensional in the Iris flower dataset is only two dimensions. The quality of the local structure in the PCA output will certainly not be as good as the t-SNE output in general.

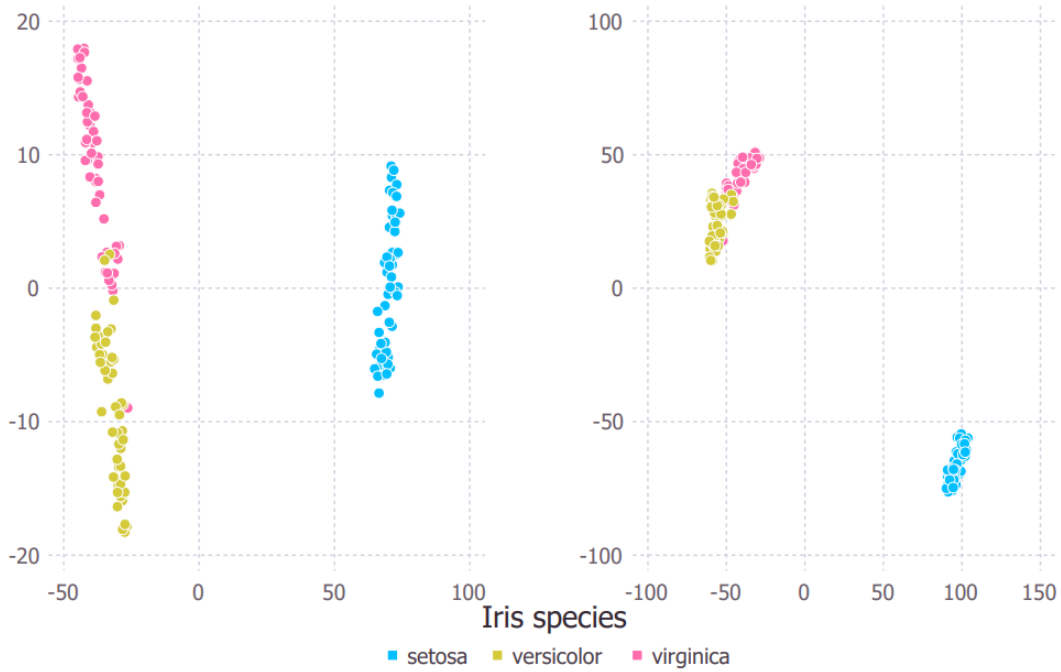

Figure 3. T-SNE outputs of the benchmark implementation (left) and my implementation (right)

Table 1. Evaluation results for the Iris flower dataset

Method	μ_{local}	SE_{local}
PCA	74.73%	13.98%
Benchmark t-SNE	77.53%	14.19%
T-SNE implementation	**78.33**%	12.55%

5.2 Evaluation results

It is important to determine the optimal perplexity value first, before evaluating the effect of the proposed extensions. Table 2 contains the results of using t-SNE to reduce the dimensionality of the word dataset from 300 to 2. A bold value indicates the best value of a column, where the local structure metric considers the sets with the 10 closest points. It can be seen in the table that the Kullback-Leibler divergence is generally lower for higher perplexity values. Consequently, it is a good idea to determine the optimal perplexity value by looking at the structural quality metrics. The quality of the local structure is robust for different perplexity values. The two embeddings with perplexity values of respectively 25 and 40 arguably perform best. The embedding with a perplexity value of 40 has the lowest Kullback-Leibler divergence and the best local structure, whereas the embedding with a perplexity value of 25 has the best global structure and the second best local structure. The standard errors of the evaluation metrics do not vary greatly.

Table 2. Evaluation results of the word representation dataset for various perplexity values

Perplexity	KL-divergence	μ_{local}	SE_{local}	μ_{global}	SE_{global}
5	0.5583	55.06%	16.91%	46.21%	21.08%
10	0.5595	56.26%	17.14%	51.77%	20.30%
15	0.5551	57.10%	17.13%	51.75%	21.02%
20	0.5278	57.62%	17.24%	51.52%	21.27%
25	0.5011	58.55%	17.68%	**53.19%**	20.02%
30	0.5079	58.02%	17.61%	49.26%	21.38%
35	0.4786	58.20%	17.58%	48.44%	22.18%
40	**0.4780**	**58.92%**	17.55%	52.03%	19.59%
45	0.4868	57.37%	18.24%	48.45%	20.42%
50	0.5036	58.17%	17.78%	52.13%	21.09%

A visual representation of the embedding with a perplexity value of 25 can be seen in Figure 4. It can be seen there that t-SNE is able to separate the different word groups fairly well. Some categories have a fair bit of overlap. For example, the category *drinks* is intertwined with some *food* word representations. The two largest categories, *capitals* and *countries*, form a big separate cluster, which seems to contain some subsclusters. A closer inspection tells us that these subsclusters correspond the to the topographic location of the words represented. This is an indication that it is perhaps better to specify these labels by continent. There are also some points that repel almost all other points. The embedding is generated in about 5 minutes.

Figure 4. T-SNE output of the word representation dataset, global perplexity = 25

Table 3 contains the same type of evaluation results as Table 2, but now with the inclusion of the proposed extension of local perplexity scaling. It can be seen in Table 3 that both the local and global structural quality become more robust for different global perplexity values. Moreover, the introduction of local perplexity scaling increases the quality of the local structure retained for all evaluated global perplexity values. Most importantly, the quality of the best embedding is about three percentage points higher than before. The quality of the global structure also improves, especially for embeddings with lower global perplexity values. The results of Table 3 can be compared with the performance of the first two principal components. The PCA embedding has a local structure performance of about 23.4% and a global structure performance of about 50.9%. As such, the t-SNE output significantly outperforms PCA on the quality of the local structure. A visual representation of an embedding generated with local perplexity scaling will be discussed later on in this subsection. Last, it is important to note that the values of the Kullback-Leibler divergence cannot directly be compared between tables, as adjusting the perplexities will always change the Kullback-Leibler divergence.

Table 3. Results for various perplexity values
*Perplexity is now complemented by local perplexity

Perplexity*	KL-divergence	μ_{local}	SE_{local}	μ_{global}	SE_{global}
5	0.8192	59.80%	16.65%	52.04%	20.08%
10	0.7948	60.68%	17.07%	**54.02**%	20.22%
15	0.7838	60.05%	17.27%	53.11%	19.40%
20	0.7758	60.40%	17.77%	53.31%	20.41%
25	0.7234	61.43%	17.34%	53.05%	19.87%
30	0.7175	61.18%	17.53%	53.33%	19.27%
35	0.6976	**61.85**%	17.25%	53.16%	21.13%
40	**0.6898**	61.26%	16.82%	52.32%	18.20%
45	0.7132	61.41%	17.64%	49.50%	22.58%
50	0.6954	61.33%	17.69%	51.25%	21.69%

The second proposed extensions gives a method for inserting new points into an existing t-SNE embedding. Local perplexity scaling will be used in the evaluation of the second extension, as this feature improves the structural quality of the embedding. The second extension can be evaluated by first considering a small collection of words that directly correspond to one of the manually created labels. This collection consists of the following five words: (1) *econometrics*, (2) *hippo*, (3) *Mexico*, (4) *parent* and (5) *statistician*. Table 4 shows the evaluation results for inserting the vector representations of these words into an existing t-SNE embedding (perplexity value of 35), where the local structure metric is based on the sets with the 25 closest points. An interesting finding is that the similarity weighted geometric median already provides good results with a moderately large power, such that the effective neighbour selection method does not need to be used explicitly. The local structure metric of these points is slightly better overall than the average local structure given in Table 3, whereas the global structure metric is somewhat worse overall.

Table 4. Evaluation results for five newly inserted words

Inserted word	*econometrics*	*hippo*	*Mexico*	*parent*	*statistician*
Local structure	64%	64%	72%	60%	52%
Global structure	28.24%	49.99%	62.52%	45.30%	60.09%

The placement of these five words can also be evaluated by looking at the encircled points in the graph of Figure 5. The words *econometrics*, *hippo*, *Mexico* and *parent* are all placed in the correct clusters of *university*, *animal*, *country* and *family* respectively. Only the word *statistician* is not inserted into a single well-defined cluster, as it is placed in between the two clusters of *university* and *jobs*. This is perhaps due to the fact that most words of the category *jobs* are also drawn to other clusters. Consequently, this might actually not be that problematic as more words seem to have this complication. Figure 5 also shows that local perplexity scaling arguably gives better defined clusters than the original t-SNE in a visual sense. This is most likely due to the fact that each point considers a more refined amount of neighbours at each iteration step.

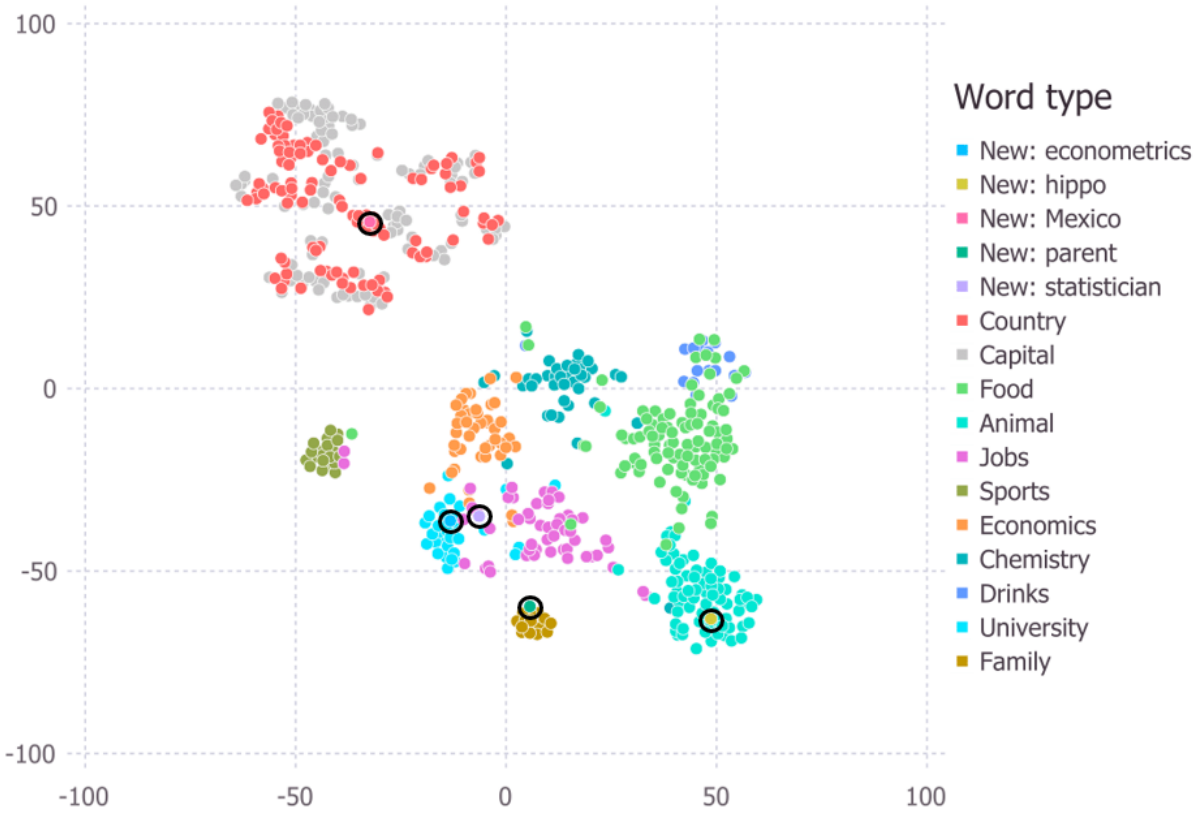

Figure 5. Newly inserted words (encircled) in an existing embedding, global perplexity = 35

The more practical application of inserting possibly unrelated words in an embedding can also be evaluated. For this purpose, fifty randomly selected nouns are inserted into the embedding according to the same principles as the five previously inserted words. It is not known if these words can be placed in one of the manually assigned categories as the words are randomly

selected. Of particular interest is the difference in results between using the similarity weighted geometric median on the test points and using t-SNE on the entire expanded dataset. The results of inserting these points according to both methods can be seen in Table 5. It can be seen there that t-SNE performs a fair amount better on the local structure metric, whereas the similarity weighted geometric median performs a slight bit better on the global structure metric. The difference in performance is primarily caused by the fact that t-SNE is able isolate points that are very dissimilar to other points. Stated differently, the proposed method will always place a point closely to its most similar words, even if these words are in fact very dissimilar. This can be addressed by giving a warning when a to be inserted point is too dissimilar from all points currently present in the embedding. Nevertheless, the primary advantage of using the proposed similarity weighted geometric median is that it takes almost no computation time because it is performed on the low-dimensional embedding. This means that it is practically independent of the number of observations and dimensions, unlike t-SNE. As such, the second proposed method is especially useful in practice.

Table 5. Average structural quality of 50 added words for the proposed method and rerunning t-SNE

	μ_{local}	SE_{local}	μ_{global}	SE_{global}
Proposed method	30.80%	20.35%	**28.43%**	18.53%
Rerunning t-SNE	**36.72%**	17.92%	27.76%	18.75%

6 Conclusion

In this research paper, I proposed two different extensions for the t-SNE algorithm based on the similarities between word representations. This was done by first presenting some relevant published papers, which was followed by a discussion of the dataset. Afterwards, two extensions for the t-SNE method were introduced, discussed and evaluated. The first extension is called local perplexity scaling, which refines the number of effective neighbours t-SNE considers at each step in the algorithm. The second extension makes use of the effective neighbour similarity weighted geometric median to efficiently and accurately place new points into an existing t-SNE embedding. Both extensions improve the quality and usability of the t-SNE method for representing textual data generated by the word2vec algorithm. In other words, the proposed extensions can be used to improve the t-SNE method for processing text via word embedding data.

The ideas presented in this paper can be more thoroughly researched in future work. The idea of inserting points in an embedding can be extended by investigating how completely dissimilar word can still be meaningfully placed in a t-SNE embedding. For example, this can potentially be done by first detecting whether a point is too dissimilar and then deciding how a certain dissimilarity measure can be incorporated in that case. The idea of local perplexity scaling can potentially be expanded by finding a more sophisticated method that differentiates the perplexities for every individual point even more faithfully.

References

Aftab, K., Hartley, R., & Trumpf, J. (2014). Generalized Weiszfeld algorithms for lq optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(4), 728–745.

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. doi: 10.1137/141000671

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

Chan, D. M., Rao, R., Huang, F., & Canny, J. F. (2019). GPU accelerated t-distributed stochastic neighbor embedding. *Journal of Parallel and Distributed Computing*, *131*, 1–13.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, *7*(2), 179–188.

Google. (2013). *Word2vec Google news model.* https://code.google.com/archive/p/word2vec/.

Hinton, G. E., & Roweis, S. T. (2003). Stochastic neighbor embedding. In *Advances in neural information processing systems 15* (pp. 857–864). MIT Press.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, *24*(6), 417.

Kobak, D., & Berens, P. (2019). The art of using t-SNE for single-cell transcriptomics. *Nature communications*, *10*(1), 1–14.

Lee, J. A., Peluffo-Ordóñez, D. H., & Verleysen, M. (2015). Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure. *Neurocomputing*, *169*, 246–261.

Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S., & Kluger, Y. (2017). Efficient algorithms for t-distributed stochastic neighborhood embedding. *arXiv preprint arXiv:1712.09005*.

Linderman, G. C., & Steinerberger, S. (2019). Clustering with t-SNE, provably. *SIAM Journal on Mathematics of Data Science*, *1*(2), 313–332.

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559-572. doi: 10.1080/14786440109462720

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Pezzotti, N., Mordvintsev, A., Hollt, T., Lelieveldt, B. P. F., Eisemann, E., & Vilanova, A. (2018). Linear t-SNE optimization for the web. *arXiv preprint arXiv:1805.10817*.

Schakel, A. M., & Wilson, B. J. (2015). Measuring word significance using distributed representations of words. *arXiv preprint arXiv:1508.02297*.

Van Der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, *15*(1), 3221–3245.

A Appendix A: programming code instructions

The provided code contains ten Julia files that contain various functions each. The code is designed to be modular, such that every function with a similar purpose is grouped together in a file. The Julia console requires that you change the current working environment to the correct directory. All functions can then be accessed by copying the following block of code into the Julia console:

```
include("tsne_init.jl")
include("tsne_distances.jl")
include("tsne_wordvec.jl")
include("tsne_perplexity.jl")
include("tsne_weights.jl")
include("tsne_insert.jl")
include("tsne_main.jl")
include("tsne_evaluation.jl")
include("tsne_save.jl")
include("tsne_experiments.jl")
```

The code is originally written in version 1.4.1 of Julia. Each file contains a brief overview at the top and in-line comments. Of particular interest is the *tsne_experiments.jl* file, which contains functions to directly reproduce all tables and figures of this paper. In short, the ten files provide the following functions:

tsne_init.jl contains different initialization options for the t-SNE algorithm.
tsne_distances.jl contains various (pseudo-)distance and similarity metrics.
tsne_wordvec.jl contains some word embedding related functions.
tsne_perplexity.jl contains local perplexity functions that can be used in t-SNE.
tsne_weights.jl contains the discussed similarity weighing functions and solvers.
tsne_insert.jl contains the discussed method for inserting new points in t-SNE.
tsne_main.jl contains the main t-SNE algorithm and directly related functions.
tsne_evaluation.jl contains functions for evaluation metrics.
tsne_save.jl contains options to save and report the t-SNE output.
tsne_experiments.jl contains functions for generating the figures and tables of this paper.

B Appendix B: composition of the dataset used

The dataset consists of eleven word categories that contain a total of 601 words. The file *thesis_words.csv* contains the words with corresponding labels and can be loaded through: *words, labels = import_words()*. The word embeddings of these terms can be obtained directly with the function *get_embeddings(words)* after downloading and compiling the code. The code also contains a file called *test_words.csv* that contains the 50 words that are used for evaluating the point insertion method

The following eleven different word categories are included:

- *Countries* (105 word representations) contains words like *Greece* and *Japan*.
- *Capitals* (105 word representations) contains words like *Rome* and *Stockholm*.
- *Drinks* (17 word representations) contains words like *coffee* and *water*.
- *Animals* (75 word representations) contains words like *duck* and *fox*.
- *Food* (96 word representations) contains words like *cheese* and *meat*.
- *Jobs* (52 word representations) contains words like *actress* and *nurse*.
- *Family* (17 word representations) contains words like *father* and *sister*.
- *Sports* (20 word representations) contains words like *baseball* and *tennis*.
- *University subjects* (32 word representations) contains words like *biology* and *philosophy*.
- *Chemistry* (39 word representations) contains words like *catalyst* and *hydrolysis*.
- *Economics* (43 word representations) contains words like *budget* and *stock*.