Supplementary Material to "Dynamic Word Embeddings"

Robert Bamler [1] **Stephan Mandt** [1]

Table 1. Hyperparameters for skip-gram filtering and skip-gram smoothing.

PARAMETER	COMMENT
$L = 10^4$	vocabulary size
$L' = 10^3$	batch size for smoothing
$d = 100$	embedding dimension for SoU and Twitter
$d = 200$	embedding dimension for Google books
$N_{\text{tr}} = 5000$	number of training steps for each t (filtering)
$N'_{\text{tr}} = 5000$	number of pretraining steps with minibatch sampling (smoothing; see Algorithm 2)
$N_{\text{tr}} = 1000$	number of training steps without minibatch sampling (smoothing; see Algorithm 2)
$c_{\text{max}} = 4$	context window size for positive examples
$\eta = 1$	ratio of negative to positive examples
$\gamma = 0.75$	context exponent for negative examples
$D = 10^{-3}$	diffusion const. per year (Google books & SoU)
$D = 1$	diffusion const. per year (Twitter)
$\sigma_0^2 = 1$	variance of overall prior
$\alpha = 10^{-2}$	learning rate (filtering)
$\alpha' = 10^{-2}$	learning rate during minibatch phase (smoothing)
$\alpha = 10^{-3}$	learning rate after minibatch phase (smoothing)
$\beta_1 = 0.9$	decay rate of 1st moment estimate
$\beta_2 = 0.99$	decay rate of 2nd moment estimate (filtering)
$\beta_2 = 0.999$	decay rate of 2nd moment estimate (smoothing)
$\delta = 10^{-8}$	regularizer of Adam optimizer

1. Dimensionality Reduction in Figure 1

To create the word-clouds in Figure 1 of the main text we mapped the fitted word embeddings from $\mathbb{R}^d$ to the two-dimensional plane using dynamic t-SNE (Rauber et al., 2016). Dynamic t-SNE is a non-parametric dimensionality reduction algorithm for sequential data. The algorithm finds a projection to a lower dimension by solving a non-convex optimization problem that aims at preserving nearest-neighbor relations at each individual time step. In addition, projections at neighboring time steps are aligned with each other by a quadratic penalty with prefactor $\lambda \geq 0$ for sudden movements.

There is a trade-off between finding good local projections for each individual time step ($\lambda \to 0$), and finding smooth projections (large λ). Since we want to analyze the smoothness of word embedding trajectories, we want to avoid bias towards smooth projections. Unfortunately, setting $\lambda = 0$ is not an option since, in this limit, the optimization problem is invariant under independent rotations at each time, rendering trajectories in the two-dimensional projection plane meaningless. To still avoid bias towards smooth projections, we anneal λ exponentially towards zero over the course of the optimization. We start the optimizer with $\lambda = 0.01$, and we reduce λ by 5% with each training step. We run 100 optimization steps in total, so that $\lambda \approx 6 \times 10^{-6}$ at the end of the training procedure. We used the open-source implementation,[1] set the target perplexities to 200, and used default values for all other parameters.

2. Hyperparemeters and Construction of $n^{\pm}_{1:T}$

Table 1 lists the hyperparameters used in our experiments. For the Google books corpus, we used the same context window size c_{max} and embedding dimension d as in (Kim et al., 2014). We reduced d for the SoU and Twitter corpora to avoid overfitting to these much smaller data sets.

In constrast to word2vec, we construct our positive and negative count matrices $n^{\pm}_{ij,t}$ deterministically in a preprocessing step. As detailed below, this is done such that it resembles as closely as possible the stochastic approach in word2vec (Mikolov et al., 2013). In every update step, word2vec stochastically samples a context window size uniformly in an interval $[1, \cdots, c_{max}]$, thus the context size fluctuates and nearby words appear more often in the same context than words that are far apart from each other in the sentence. We follow a deterministic scheme that results in similar statistics. For each pair of words (w_1, w_2) in a given sentence, we increase the counts $n^{+}_{i_{w_1} j_{w_2}}$ by $\max(0, 1 - k/c_{\text{max}})$, where $0 \leq k \leq c_{max}$ is the number of words that appear between w_1 and w_2, and i_{w_1} and j_{w_2} are the words' unique indices in the vocabulary.

[1]Disney Research, 4720 Forbes Avenue, Pittsburgh, PA 15213, USA. Correspondence to: Robert Bamler <Robert.Bamler@disneyresearch.com>, Stephan Mandt <Stephan.Mandt@disneyresearch.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

[1] https://github.com/paulorauber/thesne

Algorithm 1 Skip-gram filtering; see section 4 of the main text.

Remark: All updates are analogous for word and context vectors; we drop their indices for simplicity.
Input: number of time steps T, time stamps $\tau_{1:T}$, positive and negative examples $n^{\pm}_{1:T}$, hyperparameters.
Init. prior means $\tilde{\mu}_{ik,1} \leftarrow 0$ and variances $\tilde{\Sigma}_{i,1} = I_{d\times d}$
Init. variational means $\mu_{ik,1} \leftarrow 0$ and var. $\Sigma_{i,1} = I_{d\times d}$
for $t = 1$ **to** T **do**
  if $t \neq 1$ **then**
    Update approximate Gaussian prior with params. $\tilde{\mu}_{ik,t}$ and $\tilde{\Sigma}_{i,t}$ using $\mu_{ik,t-1}$ and $\Sigma_{i,t-1}$, see Eq. 13.
  end if
  Compute entropy $\mathbb{E}_q[\log q(\cdot)]$ analytically.
  Compute expected log Gaussian prior with parameters $\tilde{\mu}_{ik,t}$ and $\tilde{\Sigma}_{k,t}$ analytically.
  Maximize $\mathcal{L}_t$ in Eq. 11, using black-box VI with the reparametrization trick.
  Obtain $\mu_{ik,t}$ and $\Sigma_{i,t}$ as outcome of the optimization.
end for

We also used a deterministic variant of word2vec to construct the negative count matrices n^-_t. In word2vec, η negative samples (i, j) are drawn for each positive sample (i, j') by drawing η independent values for j from a distribution $P'_t(j)$ defined below. We define $n^-_{ij,t}$ such that it matches the expectation value of the number of times that word2vec would sample the negative word-context pair (i, j). Specifically, we define

$$P_t(i) = \frac{\sum_{j=1}^{L} n^+_{ij,t}}{\sum_{i',j=1}^{L} n^+_{i'j,t}}, \tag{1}$$

$$P'_t(j) = \frac{\big(P_t(j)\big)^\gamma}{\sum_{j'=1}^{L} \big(P_t(j')\big)^\gamma}, \tag{2}$$

$$n^-_{ij,t} = \Bigg(\sum_{i',j'=1}^{L} n^+_{i'j',t}\Bigg)\eta P_t(i) P'_t(j). \tag{3}$$

We chose $\gamma = 0.75$ as proposed in (Mikolov et al., 2013), and we set $\eta = 1$. In practice, it is not necessary to explicitly construct the full matrices n^-_t, and it is more efficient to keep only the distributions $P_t(i)$ and $P'_t(j)$ in memory.

3. Skip-gram Filtering Algorithm

The skip-gram filtering algorithm is described in section 4 of the main text. We provide a formulation in pseudocode in Algorithm 1.

4. Skip-gram Smoothing Algorithm

In this section, we give details for the skip-gram smoothing algorithm, see section 4 of the main text. A summary is provided in Algorithm 2.

Algorithm 2 Skip-gram smoothing; see section 4. We drop indices i, j, and k for word, context, end embedding dimension, respectively, when they are clear from context.

Input: number of time steps T, time stamps $\tau_{1:T}$, word-context counts $n^+_{1:T}$, hyperparameters in Table 1
Obtain n^-_t $\forall t$ using Eqs. 1–3
Initialize $\mu_{u,1:T}, \mu_{v,1:T} \leftarrow 0$
Initialize $\nu_{u,1:T}$, $\nu_{v,1:T}$, $\omega_{u,1:T-1}$, and $\omega_{v,1:T-1}$ such that $B_u^\top B_u = B_v^\top B_v = \Pi$ (see Eqs. 5 and 11)
for $step = 1$ **to** N'_{tr} **do**
  Draw $\mathcal{I} \subset \{1, \dots, L'\}$ with $|\mathcal{I}| = L'$ uniformly
  Draw $\mathcal{J} \subset \{1, \dots, L'\}$ with $|\mathcal{J}| = L'$ uniformly
  for all $i \in \mathcal{I}$ **do**
    Draw $\epsilon^{[s]}_{ui,1:T} \sim \mathcal{N}(0, I)$
    Solve $B_{u,i} x_{ui,1:T} = \epsilon_{ui,1:T}$ for $x_{ui,1:T}$
  end for
  Obtain $x_{vj,1:T}$ by repeating last loop $\forall j \in \mathcal{J}$
  Calculate gradient estimates of $\mathcal{L}$ for minibatch $(\mathcal{I}, \mathcal{J})$ using Eqs. 10, 14, and 15
  Obtain update steps $d[\cdot]$ for all variational parameters using Adam optimizer with parameters in Table 1
  Update $\mu_{u,1:T} \leftarrow \mu_{u,1:T} + d[\mu_{u,1:T}]$, and analogously for $\mu_{v,1:T}$, $\omega_{u,1:T-1}$, and $\omega_{v,1:T-1}$
  Update $\nu_{u,1:T}$ and $\nu_{v,1:T}$ according to Eq. 18
end for
Repeat above loop for N_{tr} more steps, this time without minibatch sampling (i.e., setting $L' = L$)

Variational distribution. For now, we focus on the word embeddings, and we simplify the notation by dropping the indices for the vocabulary and embedding dimensions. The variational distribution for a single embedding dimension of a single word embedding trajectory is

$$q(u_{1:T}) = \mathcal{N}(\mu_{u,1:T}, (B_u^\top B_u)^{-1}). \tag{4}$$

Here, $\mu_{u,1:T}$ is the vector of mean values, and B_u is the Cholesky decomposition of the precision matrix. We restrict the latter to be bidiagonal,

$$B_u = \begin{pmatrix} \nu_{u,1} & \omega_{u,1} & & & \\ & \nu_{u,2} & \omega_{u,2} & & \\ & & \ddots & \ddots & \\ & & & \nu_{u,T-1} & \omega_{u,T-1} \\ & & & & \nu_T \end{pmatrix} \tag{5}$$

with $\nu_{u,t} > 0$ for all $t \in \{1, \dots, T\}$. The variational parameters are $\mu_{u,1:T}$, $\nu_{u,1:T}$, and $\omega_{1:T-1}$. The variational distribution of the context embedding trajectories $v_{1:T}$ has the same structure.

With the above form of B_u, the variational distribution is a Gaussian with an arbitrary tridiagonal symmetric precision matrix $B_u^\top B_u$. We chose this variational distribution because it is the exact posterior of a hidden time-series model with a Kalman filtering prior and Gaussian noise (Blei & Lafferty, 2006). Note that our variational distribution is a generalization of a fully factorized (mean-field) distribution, which is obtained for $\omega_{u,t} = 0\ \forall t$. In the general case, $\omega_{u,t} \neq 0$, the variational distribution can capture correlations between all time steps, with a dense covariance matrix $(B_u^\top B_u)^{-1}$.

Gradient estimation. The skip-gram smoothing algorithm uses stochastic gradient ascent to find the variational parameters that maximize the ELBO,

$$\mathcal{L} = \mathbb{E}_q\big[\log p(U_{1:T}, V_{1:T}, n^{\pm}_{1:T})\big] - \mathbb{E}_q\big[\log q(U_{1:T}, V_{1:T})\big]. \tag{6}$$

Here, the second term is the entropy, which can be evaluated analytically. We obtain for each component in vocabulary and embedding space,

$$-\mathbb{E}_q[\log q(u_{1:T})] = -\sum_t \log(\nu_{u,t}) + \text{const.} \tag{7}$$

and analogously for $-E_q[\log q(v_{1:T})]$.

The first term on the right-hand side of Eq. 6 cannot be evaluated analytically. We approximate its gradient by sampling from q using the reparameterization trick (Kingma & Welling, 2014; Rezende et al., 2014). A naive calculation would require $\Omega(T^2)$ computing time since the derivatives of $\mathcal{L}$ with respect to $\nu_{u,t}$ and $\omega_{u,t}$ for each t depend on the count matrices $n^{\pm}_{t'}$ of all t'. However, as we show next, there is a more efficient way to obtain all gradient estimates in $\Theta(T)$ time.

We focus again on a single dimension of a single word embedding trajectory $u_{1:T}$, and we drop the indices i and k. We draw S independent samples $u^{[s]}_{1:T}$ with $s \in \{1, \dots, S\}$ from $q(u_{1:T})$ by parameterizing

$$u^{[s]}_{1:T} = \mu_{u,1:T} + x^{[s]}_{u,1:T} \tag{8}$$

with

$$x^{[s]}_{u,1:T} = B_u^{-1} \epsilon^{[s]}_{u,1:T} \quad \text{where} \quad \epsilon^{[s]}_{u,1:T} \sim \mathcal{N}(0, I). \tag{9}$$

We obtain $x^{[s]}_{u,1:T}$ in $\Theta(T)$ time by solving the bidiagonal linear system $B_u x^{[s]}_{u,1:T} = \epsilon^{[s]}_{u,1:T}$. Samples $v^{[s]}_{1:T}$ for the context embedding trajectories are obtained analogously. Our implementation uses $S = 1$, i.e., we draw only a single sample per training step. Averaging over several samples is done implicitly since we calculate the update steps using the Adam optimizer (Kingma & Ba, 2014), which effectively averages over several gradient estimates in its first moment estimate.

The derivatives of $\mathcal{L}$ with respect to $\mu_{u,1:T}$ can be obtained using Eq. 8 and the chain rule. We find

$$\frac{\partial \mathcal{L}}{\partial \mu_{u,1:T}} \approx \frac{1}{S} \sum_{s=1}^{S} \left[\Gamma^{[s]}_{u,1:T} - \Pi u^{[s]}_{1:T}\right]. \tag{10}$$

Here, $\Pi \in \mathbb{R}^{T \times T}$ is the precision matrix of the prior $u_{1:T} \sim \mathcal{N}(0, \Pi^{-1})$. It is tridiagonal and therefore the matrix-multiplication $\Pi u^{[s]}_{1:T}$ can be carried out efficiently. The non-zero matrix elements of Π are

$$\begin{aligned}
\Pi_{11} &= \sigma_0^{-2} + \sigma_1^{-2} \\
\Pi_{TT} &= \sigma_0^{-2} + \sigma_{T-1}^{-2} \\
\Pi_{tt} &= \sigma_0^{-2} + \sigma_{t-1}^{-2} + \sigma_t^{-2} \quad \forall t \in \{2, \dots, T-1\} \\
\Pi_{1,t+1} &= \Pi_{t+1,1} = -\sigma_t^{-2}.
\end{aligned} \tag{11}$$

The term $\Gamma^{[s]}_{u,1:T}$ on the right-hand side of Eq. 10 comes from the expectation value of the log-likelihood under q. It is given by

$$\Gamma^{[s]}_{ui,t} = \sum_{j=1}^{L} \left[\left(n^{+}_{ij,t} + n^{-}_{ij,t}\right) \sigma\left(-u^{[s]\top}_{i,t} v^{[s]}_{j,t}\right) - n^{-}_{ij,t}\right] v^{[s]}_{j,t} \tag{12}$$

where we temporarily restored the indices i and j for words and contexts, respectively. In deriving Eq. 12, we used the relations $\partial \log \sigma(x)/\partial x = \sigma(-x)$ and $\sigma(-x) = 1 - \sigma(x)$.

The derivatives of $\mathcal{L}$ with respect to $\nu_{u,t}$ and $\omega_{u,t}$ are more intricate. Using the parameterization in Eqs. 8–9, the derivatives are functions of $\partial B_u^{-1}/\partial \nu_t$ and $\partial B_u^{-1}/\partial \omega_t$, respectively, where B_u^{-1} is a dense (upper triangular) $T \times T$ matrix. An efficient way to obtain these derivatives is to use the relation

$$\frac{\partial B_u^{-1}}{\partial \nu_t} = -B_u^{-1} \frac{\partial B_u}{\partial \nu_t} B_u^{-1} \tag{13}$$

and similarly for $\partial B_u^{-1}/\partial \omega_t$. Using this relation and Eqs. 8–9, we obtain the gradient estimates

$$\frac{\partial \mathcal{L}}{\partial \nu_{u,t}} \approx -\frac{1}{S} \sum_{s=1}^{S} y^{[s]}_{u,t} x^{[s]}_{u,t} - \frac{1}{\nu_{u,t}}, \tag{14}$$

$$\frac{\partial \mathcal{L}}{\partial \omega_{u,t}} \approx -\frac{1}{S} \sum_{s=1}^{S} y^{[s]}_{u,t} x^{[s]}_{u,t+1}. \tag{15}$$

The second term on the right-hand side of Eq. 14 is the derivative of the entropy, Eq. 7, and

$$y^{[s]}_{u,1:T} = (B_u^\top)^{-1} \left[\Gamma^{[s]}_{u,1:T} - \Pi u^{[s]}_{1:T}\right]. \tag{16}$$

The values $y^{[s]}_{u,1:T}$ can again be obtained in $\Theta(T)$ time by bringing $B_u^\top$ to the left-hand side and solving the corresponding bidiagonal linear system of equations.

Sampling in vocabulary space. In the above paragraph, we described an efficient strategy to obtain gradient estimates in only $\Theta(T)$ time. However, the gradient estimation scales quadratic in the vocabulary size L because all L^2 elements of the positive count matrices n_t^+ contribute to the gradients. In order speed up the optimization, we pretrain the model using a minibatch of size $L' < L$ in vocabulary space as explained below. The computational complexity of a single training step in this setup scales as $(L')^2$ rather than L^2. After $N'_{\text{tr}} = 5000$ training steps with minibatch size L', we switch to the full batch size of L and train the model for another $N_{\text{tr}} = 1000$ steps.

The subsampling in vocabulary space works as follows. In each training step, we independently draw a set $\mathcal{I}$ of L' random distinct words and a set $\mathcal{J}$ of L' random distinct contexts from a uniform probability over the vocabulary. We then estimate the gradients of $\mathcal{L}$ with respect to only the variational parameters that correspond to words $i \in \mathcal{I}$ and contexts $j \in \mathcal{J}$. This is possible because both the prior of our dynamic skip-gram model and the variational distribution factorize in the vocabulary space. The likelihood of the model, however, does not factorize. This affects only the definition of $\Gamma^{[s]}_{uik,t}$ in Eq. 12. We replace $\Gamma^{[s]}_{uik,t}$ by an estimate $\Gamma^{[s]\prime}_{uik,t}$ based on only the contexts $j \in \mathcal{J}$ in the current minibatch,

$$\Gamma^{[s]}_{ui,t} = \frac{L}{L'} \sum_{j \in \mathcal{J}} \Big[\left(n^+_{ij,t} + n^-_{ij,t}\right) \sigma\left(-u^{[s]\top}_{i,t} v^{[s]}_{j,t}\right) - n^-_{ij,t} \Big] v^{[s]}_{j,t}. \tag{17}$$

Here, the prefactor L/L' restores the correct ratio between evidence and prior knowledge (Hoffman et al., 2013).

Enforcing positive definiteness. We update the variational parameters using stochastic gradient ascent with the Adam optimizer (Kingma & Ba, 2014). The parameters $\nu_{u,1:T}$ are the eigenvalues of the matrix B_u, which is the Cholesky decomposition of the precision matrix of q. Therefore, $\nu_{u,t}$ has to be positive for all $t \in \{1, \dots, T\}$. We use mirror ascent (Ben-Tal et al., 2001; Beck & Teboulle, 2003) to enforce $\nu_{u,t} > 0$. Specifically, we update ν_t to a new value ν'_t defined by

$$\nu'_{u,t} = \frac{1}{2}\nu_{u,t} d[\nu_{u,t}] + \sqrt{\left(\frac{1}{2}\nu_{u,t} d[\nu_{u,t}]\right)^2 + \nu^2_{u,t}} \tag{18}$$

where $d[\nu_{u,t}]$ is the step size obtained from the Adam optimizer. Eq. 18 can be derived from the general mirror ascent update rule $\Phi'(\nu'_{u,t}) = \Phi'(\nu_{u,t}) + d[\nu_{u,t}]$ with the mirror map $\Phi : x \mapsto -c_1 \log(x) + c_2 x^2/2$, where we set the parameters to $c_1 = \nu_{u,t}$ and $c_2 = 1/\nu_{u,t}$ for dimensional reasons. The update step in Eq. 18 increases (decreases) $\nu_{u,t}$ for positive (negative) $d[\nu_{u,t}]$, while always keeping its value positive.

Natural basis. As a final remark, let us discuss an optional extension to the skip-gram smoothing algorithm that converges in less training steps. This extension only increases the efficiency of the algorithm. It does not change the underlying model or the choice of variational distribution. Observe that the prior of the dynamic skip-gram model connects only neighboring time-steps with each other. Therefore, the gradient of $\mathcal{L}$ with respect to $\mu_{u,t}$ depends only on the values of $\mu_{u,t-1}$ and $\mu_{u,t+1}$. A naive implementation of gradient ascent would thus require $T-1$ update steps until a change of $\mu_{u,1}$ affects updates of $\mu_{u,T}$.

This problem can be avoided with a change of basis from $\mu_{u,1:T}$ to new parameters $\rho_{u,1:T}$,

$$\mu_{u,1:T} = A \rho_{u,1:T} \tag{19}$$

with an appropriately chosen invertible matrix $A \in \mathbb{R}^{T \times T}$. Derivatives of $\mathcal{L}$ with respect to $\rho_{u,1:T}$ are given by the chain rule, $\partial\mathcal{L}/\partial\rho_{u,1:T} = (\partial\mathcal{L}/\partial\mu_{u,1:T})A$. A natural (but inefficient) choice for A is to stack the eigenvectors of the prior precision matrix Π, see Eq. 11, into a matrix. The eigenvectors of Π are the Fourier modes of the Kalman filtering prior (with a regularization due to σ_0). Therefore, there is a component $\rho_{u,t}$ that corresponds to the zero-mode of Π, and this component learns an average word embedding over all times. Higher modes correspond to changes of the embedding vector over time. A single update to the zero immediately affects all elements of $\mu_{u,1:T}$, and therefore changes the word embeddings at all time steps. Thus, information propagates quickly along the time dimension. The downside of this choice for A is that the transformation in Eq. 19 has complexity $\Omega(T^2)$, which makes update steps slow.

As a compromise between efficiency and a natural basis, we propose to set A in Eq. 19 to the Cholesky decomposition of the prior covariance matrix $\Pi^{-1} \equiv AA^\top$. Thus, A is still a dense (upper triangular) matrix, and, in our experiments, updates to the last component $\rho_{u,T}$ affect all components of $\mu_{u,1:T}$ in an approximately equal amount. Since Π is tridiagonal, the inverse of A is bidiagonal, and Eq. 19 can be evaluated in $\Theta(T)$ time by solving $A\mu_{u,1:T} = \rho_{u,1:T}$ for $\mu_{u,1:T}$. This is the parameterization we used in our implementation of the skip-gram smoothing algorithm.

References

Beck, Amir and Teboulle, Marc. Mirror Descent and Nonlinear Projected Subgradient Methods for Convex Opti-

mization. *Operations Research Letters*, 31(3):167–175, 2003.

Ben-Tal, Aharon, Margalit, Tamar, and Nemirovski, Arkadi. The Ordered Subsets Mirror Descent Optimization Method with Applications to Tomography. *SIAM Journal on Optimization*, 12(1):79–108, 2001.

Blei, David M and Lafferty, John D. Dynamic Topic Models. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 113–120. ACM, 2006.

Hoffman, Matthew D, Blei, David M, Wang, Chong, and Paisley, John William. Stochastic Variational Inference. *Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Kim, Yoon, Chiu, Yi-I, Hanaki, Kentaro, Hegde, Darshan, and Petrov, Slav. Temporal Analysis of Language Through Neural Language Models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pp. 61–65, 2014.

Kingma, Diederik and Ba, Jimmy. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik P and Welling, Max. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. 2013.

Rauber, Paulo E., Falcão, Alexandre X., and Telea, Alexandru C. Visualizing Time-Dependent Data Using Dynamic t-SNE. In *EuroVis 2016 - Short Papers*, 2016.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *The 31st International Conference on Machine Learning (ICML)*, 2014.