Text Style Transfer Using Partly-Shared Decoder

Gil Halevi
Briarcliff High School
Briarcliff Manor, United States
halevigh@gmail.com

Kahini Wadhawan
IBM Research
Yorktown Heights, United States
kahini.wadhawan@ibm.com

***Abstract*—This work expands upon existing approaches to text style transfer, the task of changing the style of a sentence without changing its content. This task could have many applications, from creating headlines for a general audience based on scientific paper headlines to making offensive language more suitable for children. We focus specifically on sentiment transfer, due to the availability of large sets of data. Text style transfer is uniquely difficult since it is hard to find parallel data, sentence-for-sentence translations, to train on. In this work, we begin by training an encoder neural network to produce representations of text that encode only content, not style, using an adversarial approach. We then introduce a new partly-shared decoder architecture to turn these representations back into sentences, attempting to achieve a better mix of content preservation and style modification. We find that a partly-shared decoder results in content preservation and style modification scores in between different baselines, suggesting a potential utility depending on the desired application.**

I. Introduction

Neural text style transfer has been gaining interest recently in the Natural Language Processing and Artificial Intelligence community. This task involves taking text of a certain style and transferring it to another style without altering its content using neural networks. Style transfer with sentiment, for example, would involve taking the positive sentence "I love the store" and transferring it into a negative sentence "I hate the store". Sentiment is the most common style transfer task due to the availability of data obtained from review sites, and it is the focus of our work. Other uses of style transfer that have been explored include transferring academic paper titles to news article titles [1], which could aid in writing academic news reports based on scientific advancements, and making offensive text non-offensive, which could help make offensive text online more suitable for children . [2].

One challenge in style transfer is that there is a tradeoff between how well a model can modify the style and preserve the content of sentences. One factor controlling this is how sentences are produced. A common method in style transfer is to have each sentence encoded into a representation that encodes just its contents and not its style. A model called the decoder then turns this representation back into a sentence. Most researchers use a single decoder for both styles, with an indicator telling the decoder which style to decode each sentence into. Reference [1] attempt instead to have a separate decoder for each style, which they find to be more effective at changing the style of sentences but less effective at preserving the content of sentences.

In this paper, we propose a new decoder architecture for a style transfer model, one where the decoder is *partly* shared between the two models, in the hope that it will achieve a better mix of content preservation and style transfer. We evaluate this model compared to previous decoder architectures on sentiment transfer using the Yelp review dataset.

II. Background

Neural networks [3] learn to estimate a function by training on data, usually made of input-output pairs, and minimizing a loss function which evaluates how far the network's output differs from the ideal output. Networks work with vectors - lists of numbers. Reference [4] created the recurrent neural network (RNN), a type of neural network that receives a sequence of inputs rather than a single input. This has been useful for creating representations of sentences, as these models can take sequences of words as input.

A text autoencoder [5] is a model that learns to represent sentences as a single vector. It does this using an encoder RNN, which learns to convert a sentence into the vector, and a decoder RNN, which learns to convert the vector back into a sentence. A sequence to sequence (seq2seq) model [6], as it is applied to sentence translation, is a slight variation on a text autoencoder. In seq2seq, the vector representation is decoded not to match the original sentence, but to match the translation of this sentence into a different language.

The difficulty in text style transfer is that for most styles, large parallel corpora – where every sentence has its exact translation in another style to train on – are not available to train on. Such data are necessary for training seq2seq models. Without these data, learning to transfer the content of sentences without their style becomes much harder and requires different techniques. In this paper, we propose a new decoder architecture for an autoencoder-based style transfer model, in hopes that it will achieve a better mix of content preservation and style transfer than previous approaches.

III. Related Research

The main challenge in style transfer is creating a model that can preserve the content of a sentence while changing the style, without being given parallel data to base said transformations on. Some research [7], [8] has attempted to do sentiment transfer by training a model to explicitly change only words which exhibit sentiment. Reference [9] use this method to

generate parallel corpora, which can then be used to train a normal seq2seq model.

Many researchers, however, create more general models using autoencoders, with tweaked loss functions to ensure that style is modified but not content. These architectures have two (or more, if style transfer between more than two styles is attempted) different autoencoders, each trained on reconstructing a different style of data. To transfer sentences from one style to another, they are encoded by the encoder of their source style and decoded by the decoder of the target style. Some of these models [2], [10] are also trained on back-transfer, where sentences from one style are transferred to the second style and then back-transferred to the first style. The cyclic consistency loss for back-transfer measures the similarity between the original sentence and the back-transferred sentence.

Some researchers attempt to train the autoencoders by simply enforcing, through an added loss, that the target sentence matches the preferred style, while still maintaining reconstruction loss. Reference [11] use a collaborative Convolutional Neural Network, or CNN [12], classifier trained on the same data as the target sentences. Reference [13] use a language model instead, which can rate how natural the target sentence is in its given style.

However, much of the research regarding style transfer has focused on separating content from style for the latent vector of each sentence. By encoding a sentence as a latent vector that only reflects content, it is possible to decode it into any style without having a decoder that has been trained to modify style. Reference [14] use a separate pre-trained translation model to extract a latent vector from a sentence that reflects only content and not style, although this is not applicable to sentiment transfer.

Other researchers use adversarial models to rid the latent vectors of style. An adversarial model [15] is a model trained in parallel to the primary model, which in this case would be the autoencoders, on a different loss function. The adversarial model usually tries to learn some quality about an input given the output of the primary model. The primary model, meanwhile, tries to fool the adversarial model, making the adversarial model unable to learn this quality. Reference [10] apply this to style transfer. They train their model using back-transfer and add adversarial discriminators to differentiate between the decoder hidden states of sentences that are being reconstructed and those that are being back-transferred. The encoder is trained to fool the discriminator, and in doing so ensures that sentences from both styles are encoded in the same way. This should lead to encoded sentences that do not exhibit style. Reference [1] train an adversarial classifier to differentiate between encoded sentences of different styles, which the encoder tries to fool to similar effect.

An important distinction among different autoencoder-based style transfer models is whether the encoders and decoders of different styles have any shared parameters. Many researchers [10], [13], [14] have completely shared encoders and decoders, with a simple indicator specifying style passed in to both the encoder and decoder. Reference [1], on the other hand, use one encoder with no style indication, in order to better align the distributions. For decoders, they use two variants: one with two completely separate style-specific decoders, and another with a single decoder that takes in a full vector — not just a single indicator — specifying the characteristics of the style decoded into. The vectors representing each style are learned during training.

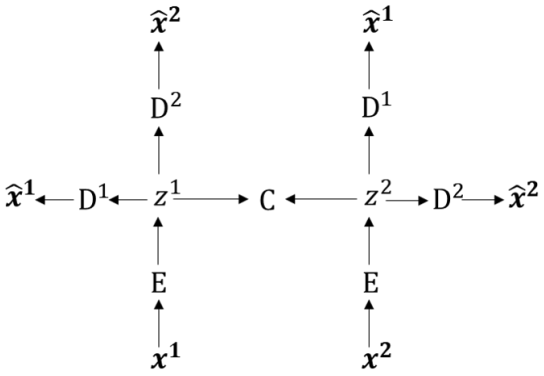

Fig. 1. An overview of the architecture of our model: x represents a sentence, E represents an encoder, z represents a latent vector, D represents a decoder, and C represents the adversarial classifier.

IV. Methods

The methods we use are based on an RNN autoencoder, which learns to represent a sentence as a vector and then reconstruct the sentence from that vector. Each sentence is represented as $\mathbf{x} = \{x_1, \ldots, x_l\}$, where x is a word vector and l is the length of the sentence. The autoencoder is made up of an encoder RNN and a decoder RNN, both of which employ the Gated Recurrent Unit (GRU) architecture [16]. The encoder learns to convert each sentence $\mathbf{x}$ into a latent vector z:

$$z = E(\mathbf{x}; \theta_E) \tag{1}$$

where θ_E are the parameters for the encoder. Given the encoders output, the decoder D learns to build a target sentence $\hat{\mathbf{x}}$ by choosing the most probable word, one word at a time, until the full sentence is produced:

$$p(\hat{\mathbf{x}}|\mathbf{x}; \theta_D) = \prod_{t=1}^{l} p(\hat{x}_t|E(\mathbf{x}; \theta_E); \hat{x}_1, \ldots, \hat{x}_{t-1}; \theta_D) \tag{2}$$

where θ_D are the parameters for the decoder. More specifically, this decoder is made up of a decoding GRU unit d and softmax layer g which produce hidden vector h at each timestep:

$$h_0 = z \tag{3}$$

$$h_n = d(h_{n-1}, x_{n-1}; \theta_D) \tag{4}$$

$$p(\hat{x}_n) = g(h_n) \tag{5}$$

The loss functions for the whole reconstruction is:

$$L_{rec}\ (\theta_E, \theta_D) = -\sum_{n=1}^{N} \log(\hat{\mathbf{x}}_n|\mathbf{x}_n; \theta_E, \theta_D) \tag{6}$$

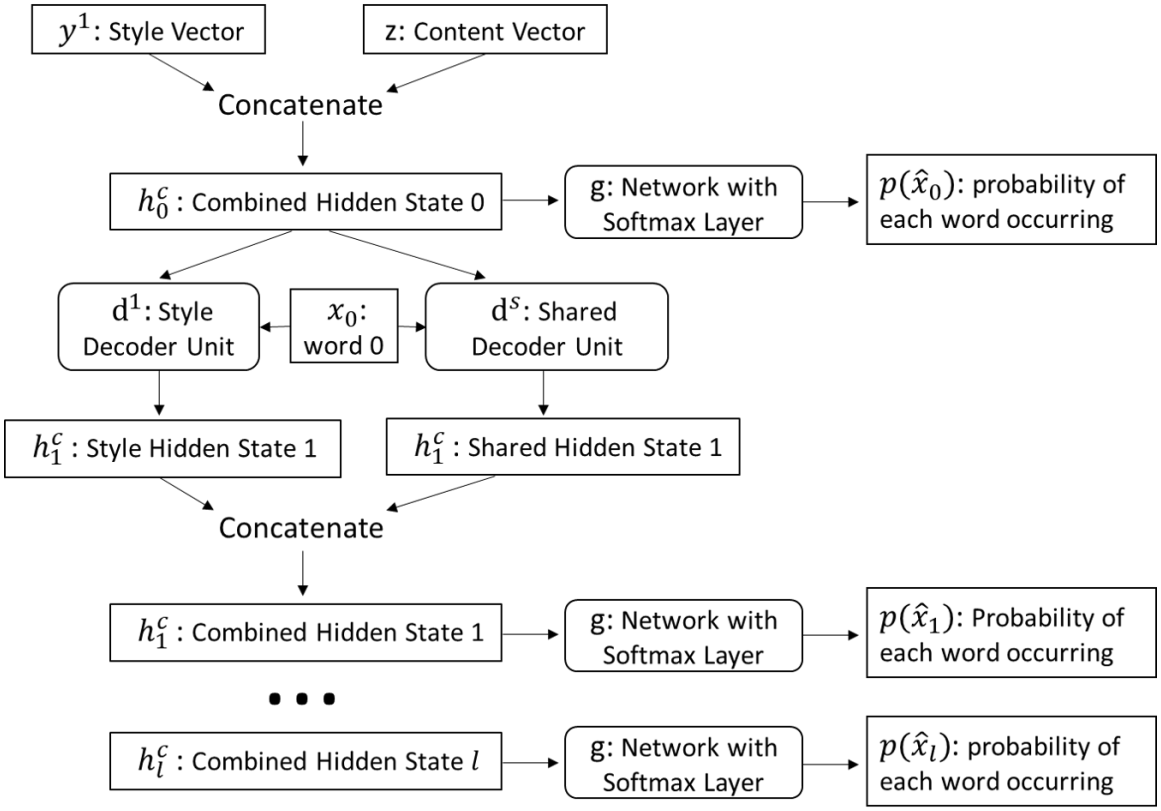

Fig. 2. The decoder architecture for decoding in s^1. The same architecture is used for decoding in s^2.

where N is the size of the dataset. We apply the autoencoder to style transfer. Given two datasets, $\mathbf{X}^1 = \{\mathbf{x}_1^1, \ldots, \mathbf{x}_N^1\}$ with style s^1 and $\mathbf{X}^2 = \{\mathbf{x}_1^2, \ldots, \mathbf{x}_N^2\}$ with style s^2, style transfer is the task of converting sentence $\mathbf{x}^1$ to $\mathbf{x}^2$ without changing the content. All our methods can also be applied to style transfer among more than two styles.

We train our style transfer model as two autoencoders. That is, we have one autoencoder designed for reconstructing $\mathbf{x}^1$, trained on sentences from $\mathbf{X}^1$, and another model designed for reconstructing $\mathbf{x}^2$, trained on sentences from $\mathbf{X}^2$. To transfer from one style to another during inference, sentences in style s^1 are encoded and then decoded using the decoder for $\mathbf{X}^2$, and vice versa for sentences in style s^2. In order to be able to decode the latent vector in a sentence into either style, we want the latent vector to represent the content of a sentence without the style. One way we accomplish this is by using one shared encoder for both styles, ensuring that content from sentences from both styles are encoded in the same distribution. Following [10], we also train an adversarial network called the classifier, C, with the goal of predicting the style of each sentence from its latent vector z. The loss for this classifier can be represented as:

$$L_{cls}(\theta_c) = -\sum_{n=1}^{N} \log \mathrm{p}(s|\mathbf{x}_n; E, c) \tag{7}$$

The encoder of our neural network is then trained on minimizing the Shannon entropy of the classifiers prediction, pushing it toward a neutral prediction. By doing this, the encoder learns to encode the content of a sentence in a vector from which style is not inferable. This adds the loss:

$$L_{adv}(\theta_E) = -\sum_{m=1}^{M}\sum_{n=1}^{N} H(p(s|\mathbf{x}_n; E, c)) \tag{8}$$

where M is the total number of styles. The basic architecture of the model is shown in fig. 1, where sentences from both styles are inputted into the same encoder and then decoded using the decoder of the target style.

As seen in fig. 2, we propose a partly-shared decoder architecture. This work applies the partly shared recurrent models from [17], though we use this only in the decoder as opposed to also in the encoder. We additionally incorporate an adversarial classifier to rid part of the latent vector of style information, something not done in [17].

Our partly-shared decoder includes a learned style embedding $E \in R^{S \ x \ d_s}$, where S is the number of styles and each style vector has size d_s. The partly-shared decoder gets fed in the z vector of size d_c representing the content of a sentence, concatenated with the style vector y for the target style picked from the embedding. Then, for each timestep of the decoder GRU, the shared part of the decoder takes the hidden state as input and outputs a vector of size d_c. The separate part of the decoder also takes the hidden state as input, and outputs

a hidden vector of size d_s. The concatenation of these two vectors then becomes the next combined hidden state, and as such is converted to a probability distribution of the next word. The following are the functions for decoder D^1:

$$h_0^c = [z, y^1] \tag{9}$$
$$h_n^1 = d(h_{n-1}^c, x_{n-1}; \theta_D^1) \tag{10}$$
$$h_n^s = d(h_{n-1}^c, x_{n-1}; \theta_D^s) \tag{11}$$
$$h_n^c = [h_n^s, h_n^1] \tag{12}$$
$$p(\hat{x}_n) = g(h_n^c) \tag{13}$$

here h_n^c is the combined hidden state, h_n^c is the output of the shared decoder cell and h_n^s is the output of the separate decoder cell. Equivalent functions are also used for D^2. The loss for reconstructing a sentence in s^1 is as follows:

$$L_{rec}\ (\theta_E, \theta_D) = -\sum_{n=1}^{N} \log p(\hat{\mathbf{x}}_n | \mathbf{x}_n; \theta_E, \theta_D^s, \theta_D^1, y^1) \tag{14}$$

The same loss is applied to s^2. The total loss is the sum of the losses, with α being a hyperparameter:

$$L_{total} = L_{adv} + \alpha L_{cls} + L_{rec}^1 + L_{rec}^2 \tag{15}$$

V. Experimentation

We train our model on the Yelp review dataset, which consists of 444101 training, 63483 validation and 126670 test set sentences, for the task of sentiment transfer. Each sentence in the dataset is classified as either positive or negative, depending on the type of review that it came from. We transfer all of the test set sentences with our model, changing the positive reviews into negative and vice versa. We use the transferred sentence to measure how well the model modifies the style and preserves the content of sentences.

Our style modification score is equal to the proportion of the transferred sentences that match the target style. This is measured by a pre-trained CNN style classifier [18], which obtains an accuracy of 97% on our test dataset. Content preservation is measured as the cosine similarity between a vector representing the source sentence and another vector representing the target sentence, which was found previously to correlate well with human judgement [10]. This sentence vector is obtained by first removing any sentiment-specific words on a stoplist [19] in order to limit the effect that changing any of these words will have on the content preservation score. Then, the sentence vector is created by min, mean and max pooling the word vectors of all of the words that make up the sentence to extract the important features of the sentence:

$$v_{min} = min(x_1, \ldots, x_l) \tag{16}$$
$$v_{mean} = mean(x_1, \ldots, x_l) \tag{17}$$
$$v_{max} = max(x_1, \ldots, x_l) \tag{18}$$
$$v_s = [v_{min}, v_{mean}, v_{max}] \tag{19}$$

The cosine similarity between vectors of the sentence before and after it is transferred is then taken as the content preservation score:

$$score = \frac{{v^1}^T \hat{v}^2}{\|v^1\| \|\hat{v}^2\|} \tag{20}$$

where $\hat{v}$ is the vector of the transferred sentence.

We use pretrained word embeddings from word2vec [20], generated using genism [21]. Our models were trained for 12 hours on a Nvidia P106. The learning rate for our encoder-decoder model was 0.001 while the learning rate for the classifier was 0.0001. The batch size for our model is 64. We updated the classifier 4 times in between each encoder-decoder update. The adversarial weight α was 100. The content vectors in our model have 128 dimensions while the style vector and separate part of our decoder have 64 dimensions.

We compare the results of this partially shared model with two other baselines, based on [10]. The shared model has one decoder that takes in a style vector. This is the same as in our model, except in the shared model, there are no separate parameters for decoding into different styles. The separate model has two completely separate decoders, one for each style, and the decoders do not take a style vector as input.

VI. Results

Our quantitative results are shown in Table 1 and some example sentences undergoing style transfer are shown in Table 2. As seen in Table 2, in some instances our partly-shared model does a significantly better job at style transfer than the two baselines. As for the quantitative result, our partly-shared model scored higher than the shared model in style modification score, but worse than the separate model. Conversely, our model outperformed the separate model in content preservation but underperformed the shared model. This makes sense, as the more parameters shared between styles, the better a model will be at preserving content due to the similar decoding in both styles. However, such improvement comes at the sacrifice of style modification, where having similar decoders for both styles will limit the ability of the model to apply the target style to a content vector. Our results suggest that this partly-shared model may work well as a compromise model, depending on the application.

The tradeoff between style modification and content preservation, however, may also partially reflect imperfect quantitative measures. Our measure of content preservation, cosine similarity, may reflect some style information even after words indicative of style are removed. This factor could be partially responsible for the tradeoff between style modification and content preservation, where sentences that match the target style better receive lower content preservation scores due to the limitations of cosine similarity.

VII. Discussion

In this paper, we experiment with a novel partly-shared decoder architecture in a style transfer model. We find that this partly-shared decoder has results that fall in between a

TABLE I
THE STYLE MODIFICATION AND CONTENT PRESERVATION OF OUR MODEL COMPARED TO BASELINES.

	Shared Baseline	Partly-Shared	Separate Baseline
Style Modification	0.385	0.447	**0.758**
Content Preservation	**0.940**	0.907	0.875

TABLE II
EXAMPLES OF STYLE TRANSFER BY OUR MODEL COMPARED TO BASELINES

Transfer Type	Original	Shared Basline	Partly-Shared	Separate Baseline
Positive to Negative	I met the owner and he is the nicest guy ever.	I met the owner and he did not deserve a bag.	I met the office owner - the service is terrible.	I called the nurse wasn't my pain she even called me.
Negative to Positive	Attention to detail is sorely lacking.	Attention to detail is outstanding service.	Attention to detail. is phenomenal	Attention to detail is skilled, the studio was seriously.

completely separate and completely shared decoder model for content preservation and style modification. This suggests that this partly-shared architecture may be more useful, depending on the application.

Future research will likely focus on new, perhaps more complex approaches than autoencoders to transfer style. More specialized methods for style transfer may also be required for different styles. Sentiment relies heavily on the word choice, such as good and evil, while other styles such as academic paper-news headline depend more on word order.

In the future, research will also focus on creating better ways to assess style transfer models. This could help us better determine the extent to which the observed tradeoff between content preservation and style transfer is a consequence of imperfect measures, rather than a necessary tradeoff with this method of style transfer.

REFERENCES

[1] Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan, "Style transfer in text: Exploration and evaluation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[2] C. N. d. Santos, I. Melnyk, and I. Padhi, "Fighting offensive language on social media with unsupervised text style transfer," *arXiv preprint arXiv:1805.07685*, 2018.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[4] A. Robinson and F. Fallside, *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering, 1987.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[10] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola, "Style transfer from non-parallel text by cross-alignment," in *Advances in Neural Information Processing Systems*, 2017, pp. 6830–6841.

[7] J. Li, R. Jia, H. He, and P. Liang, "Delete, retrieve, generate: A simple approach to sentiment and style transfer," *arXiv preprint arXiv:1804.06437*, 2018.

[8] J. Xu, X. Sun, Q. Zeng, X. Ren, X. Zhang, H. Wang, and W. Li, "Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach," *arXiv preprint arXiv:1805.05181*, 2018.

[9] Z. Zhang, S. Ren, S. Liu, J. Wang, P. Chen, M. Li, M. Zhou, and E. Chen, "Style transfer as unsupervised machine translation," *arXiv preprint arXiv:1808.07894*, 2018.

[11] I. Melnyk, C. N. d. Santos, K. Wadhawan, I. Padhi, and A. Kumar, "Improved neural text attribute transfer with non-parallel data," *arXiv preprint arXiv:1711.09395*, 2017.

[12] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, vol. 1. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988, pp. 21–28.

[13] Z. Yang, Z. Hu, C. Dyer, E. P. Xing, and T. Berg-Kirkpatrick, "Unsupervised text style transfer using language models as discriminators," *arXiv preprint arXiv:1805.11749*, 2018.

[14] S. Prabhumoye, Y. Tsvetkov, R. Salakhutdinov, and A. W. Black, "Style transfer through back-translation," *CoRR*, vol. abs/1804.09000, 2018. [Online]. Available: http://arxiv.org/abs/1804.09000

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[16] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[17] Y. Zhang, N. Ding, and R. Soricut, "Shaped: Shared-private encoder-decoder for text style adaptation," *arXiv preprint arXiv:1804.04093*, 2018.

[18] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[19] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 168–177.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[21] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer, 2010.