Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference "Dialogue 2020"

Moscow, June 17–20, 2020

RUSSIAN NATURAL LANGUAGE GENERATION: CREATION OF A LANGUAGE MODELING DATASET AND EVALUATION WITH MODERN NEURAL ARCHITECTURES

Shaheen Z. (shaheen@itmo.ru)[1, 2],
Wohlgenannt G. (gwohlg@itmo.ru)[1],
Zaity B. (bassel.zaity@gmail.com)[2],
Mouromtsev D. I. (mouromtsev@itmo.ru)[1],
Pak V. G. (vadim.pak@cit.icc.spbstu.ru)

[1]Faculty of Software Engineering and Computer Systems, ITMO University;
[2]Institute of Computer Science and Technology Peter the Great
St. Petersburg Polytechnic University (SPbPU), St. Petersburg, Russia

Generating coherent, grammatically correct, and meaningful text is very challenging, however, it is crucial to many modern NLP systems. So far, research has mostly focused on English language, for other languages both standardized datasets, as well as experiments with state-of-the-art models, are rare. In this work, we i) provide a novel reference dataset for Russian language modeling, ii) experiment with popular modern methods for text generation, namely variational autoencoders, and generative adversarial networks, which we trained on the new dataset. We evaluate the generated text regarding metrics such as perplexity, grammatical correctness and lexical diversity.

Key words: natural language generation, variational autoencoder, dataset construction, seqGAN

DOI: 10.28995/2075-7182-2020-19-644-657

1. Introduction

Text generation is a key component in many NLP systems that produce text such as translation systems, dialogue systems, or text summarization. The quality of the generated text is critical in these systems, it should be coherent and well-formed, without grammatical mistakes, and semantically meaningful [5]. Generating human-like text is challenging, it includes modeling high-level syntactic properties and features like sentiment and topic [1].

Natural Language Generation (NLG) produces human-understandable NL text in a systematic way—based on non-textual data (eg. a knowledge base) or from meaning representations (eg. a given state of a dialogue system) [16]. Modern NLG systems often make use of (neural) language models [18]. A language model (LM) is a probability distribution over a sequence of words, and can be used to predict the next word given an input sequence.

In recent years, various types of neural network architectures have been successfully applied in NLG, such as variational autoencoders (VAE) [1], [21], generative adversarial networks (GAN) [3], [5], [22], and recurrent neural networks (RNN) [11]. Here, we experiment with those architectures on Russian language.

The *goals* of this paper are (i) to create a reference dataset for language modeling for the Russian language, comparable to the popular Penn Tree Bank (PTB) dataset for English language, and (ii) to adapt and to train several state-of-the-art language models and to evaluate them on the task of Russian language text generation. We create a dataset of 236K sentences by sampling from the Lenta News dataset, preprocess the text, and filter sentences that do not match certain quality criteria. Then we train six models (four VAE models with different scheduling methods, seqGAN, and LSTM RNNLM) on the new corpus, and evaluate them regarding the perplexity metric, and manually validate 100 sentences for each model regarding grammatical correctness. We achieve best results with the VAE models, the *zero* variant performs well regarding perplexity, but overall the *cyclical* VAE model shows the highest performance, as it generates the largest fraction of grammatically correct sentences, which have similar characteristics (sentence length, etc.) as the training data.

2. Related Work

Our Russian language dataset is inspired by the plain-text/language-modeling part of the PTB dataset[1]. PTB contains about 1M words from 1989 Wall Street Journal material, with various annotations such as POS-tags. This dataset is very popular among NLP researchers for language modeling and other NLP tasks. Many recent language models are trained and evaluated also on larger corpora, such as WikiText-103 [12], or WebText [17] (created for the GPT transformer models). For languages other than English high-quality reference datasets are rare.

In language modeling, Recurrent Neural Network Language Models (RNNLM) [14], and extensions such as long short-term memory (LSTM) [19], are frequently used

[1] https://catalog.ldc.upenn.edu/LDC99T42

architectures. RNNLMs generate text word-by-word depending on a hidden state that summarizes the previous history. These models are able to capture long-range dependencies, however, they do not expose interpretable states that represent global features like topic or sentiment. Regarding recent RNNLMs, for example Merity et al. [11] investigate different strategies for regularizing and optimizing LSTM-based models.

Variational Autoencoders (VAEs) [6] have been applied to many domains, including language modeling [1], [21]. They showed impressive results in producing interpretable representations of global features like the topic or of high-level syntactic properties. For example, Yang et al. [21] use the method for unsupervised clustering of the text. VAEs are trained using regularization to avoid overfitting and produce a regular latent space that has properties enabling the generative process. Recent research on VAEs focuses on improving the quality of the hidden representation, on exploring the properties of the latent space, and experiments with different architectures to improve VAEs.

Generative adversarial networks GANs [4] train a *generator* that tries to produce realistic samples from a data distribution. The generator is guided by a *discriminator* on how to modify its parameters. GANs have been applied successfully to computer vision tasks, however, adapting GANs to generate texts is challenging due to the discrete nature of natural language. Many attempts to adopt GANs to text rely on using reinforcement learning [3], [22], or on Gumbel-Softmax approximation [9] (a continuous approximation of the softmax function). Zhang et al. [23] use a feature matching scheme for training GANs to generate realistic-looking text.

Little work exists on NLG for the Russian language. Nesterenko [15] uses a simple template-based system to generate stock market news in Russian. Kipyatkova and Karpov [7] study the use of RNNLM models in Russian speech recognition. Kuratov and Arkhipov [8] train a BERT (transformer) language model on Russian text (RuBERT) and evaluate it on tasks such as paraphrase and sentiment detection. Finally, Shimorina et al. [20] present an English-Russian parallel corpus for generating natural language text from the triples of a knowledge base (data-to-text NLG). Their corpus was created with neural machine translation. However, to the best of our knowledge, for general Russian NLG no research work has been published about general-domain NLG datasets and about the evaluation of NLG models based on modern neural architectures.

3. Variational Autoencoder

In this section, we introduce the VAE variants (zero, constant, linear, cyclical) which are applied in the experiments. An autoencoder (AE) consists of an encoder that encodes an input sequence into a hidden state and a decoder that uses this hidden state to reconstruct the original sequence. In a standard AE for language modeling, an RNN is used for both the encoder and the decoder. The decoder is then used for text generation, where each output token is conditioned on the previous output tokens. A Variational Autoencoder (VAE) encodes the input sequence x into a region in the latent space rather than a single point, this region is defined using a multi-variate Gaussian prior $p(z)$, where the last hidden state of the encoder (z) is projected on two separate vectors. These vectors represent the mean and the diagonal co-variance matrix of the prior. To restore the original sequence, the initial state of the decoder

is sampled from the prior, and then used to decode the output sequence. This way, the model is forced to be able to decode plausible sentences from every point in the latent space, that has a reasonable probability under the prior [1]. A standard recurrent neural network language model is based on a series of next-step predictions, thus a standard AE does not provide an interpretable representation of global features such as the topic or of high-level syntactic properties.

The VAE modifies the AE architecture by replacing the deterministic encoder with a learned posterior recognition model $q(z|x)$. If the VAE were trained with standard AE reconstruction objective, it would learn to encode x deterministically by making $q(z|x)$ vanishingly small. However, we want the posterior to be close to the prior (most often standard Gaussian), therefore we have two objectives and the goal is to optimize the following lower-bound:

$$L(\theta; x) = -KL(q\theta(z|x)||p(z)) + E_z \sim {}_q\theta_{(z|x)}[log\, p\theta(x|z)] \leq log\, p(x) \quad (1)$$

The first term is the KL-divergence of the posterior from the prior, and the second is the reconstruction loss, where θ stands for the parameters of the neural network. Straightforward training of the network using this objective will bring $q(z|x)$ to be exactly the same as the prior $p(z)$, and KL-divergence term in the cost function to zero. As a result, the model will behave like a standard RNNLM. Bowman et al. [1] use KL-cost annealing to solve this problem, by multiplying a variable weight β with the KL term at training time. In the beginning, β will be set to zero, and then it gets gradually increased, forcing the model to smooth out its encodings and pack them into the prior. Later research by Fu et al. [10] investigates KL annealing further, they experiment with three scheduling approaches:

- Constant Schedule: the standard approach is to keep $\beta = 1$ fixed during training, which causes the vanishing of the KL-term, the model will behave as a standard RNNLM.
- Monotonic (linear) Annealing Schedule: this is the previously described approach for VAEs by Bowman et al. [1]. It starts with $\beta = 0$ and gradually increases during training to $\beta = 1$.
- Cyclical Annealing Schedule: split the training process into M cycles, each cycle consists of two stages:
  1. Annealing, where β is annealed from 0 to 1 in the first $R[T/M]$ training steps over the cycle; R: proportion used to increase β, T: the total number of global steps, M: the number of cycles.
  2. Fixing: fix $\beta = 1$ for the rest of the cycle.

According to Fu et al. [10] cyclical KL-annealing results in better latent codes by leveraging the informative representations of previous cycles as warm re-starts.

4. SeqGAN

A Generative Adversarial Network (GAN) contains two main models: the discriminator D is trained to distinguish between real and fake data and the generator G, which is trained to fool D and tries to generate samples from the real data distribution.

D and *G* are trained simultaneously, the improvement in one model will cause further improvements in the other model. When *G* is able to produce samples from the same data distribution as the real data, *D* will be no longer able to distinguish between real and sampled data.

GANs were applied successfully to tasks in computer vision. However, adapting GANs to text generation is not straightforward, because the original GAN works with continuous data, meanwhile, text is discrete and the generator uses a non-differential function to produce each token, therefore it is difficult to pass gradient updates from *D* to *G*. Another difficulty is that *D* can evaluate only the completed text; for a partially generated sequence the evaluation score depends on the score of the full sentence.

To address these problems, Yu et al. [22] use Reinforcement learning to train GANs for text generation. The generator is treated as an agent of reinforcement learning, and the state corresponds to the generated tokens up to this moment and the action is the next token to be generated. A discriminator will give the feedback reward that guides the learning process.

We can consider *G* as a stochastic parametric policy where Monte Carlo (MC) search is used to approximate the state-action value using the policy gradient. *D* is trained by providing positive examples from the real data and negative examples from the generated samples by *G*. *G* is updated using the policy gradient and MC search by the reward signal received from *D*. The reward represents how likely the discriminator is fooled by the generator. Yu et al. [22] use a rollout policy with MC search, they set the rollout policy the same as *G* during the experiments. For *G* they choose a recurrent neural network with LSTM cells, and *D* uses a convolutional neural network with highway architecture.

The training strategy is as follows: first *G* is pre-trained using maximum likelihood estimation, then *G* and *D* are trained alternatively. To train *D*, at each training step we sample negative examples using *G* and use examples from the true data as the positive examples.

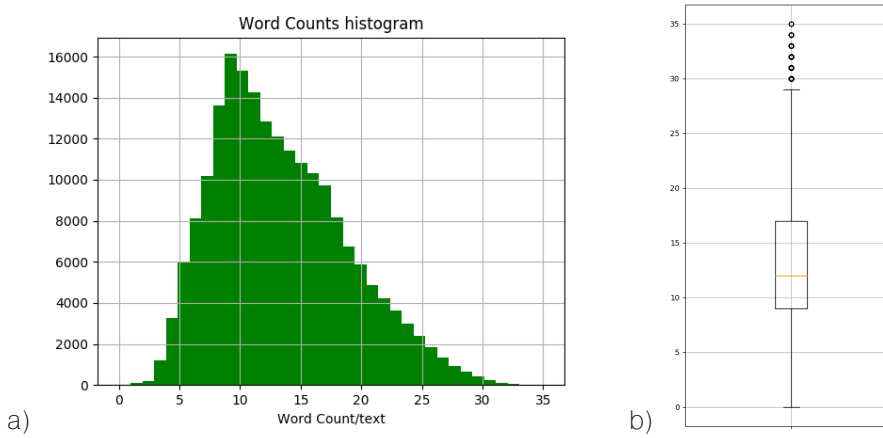

Figure 1: a) A histogram visualizing the number of words per sentence which shows the probability distribution of the data, b) A box plot for the number of words per sentence

5. Dataset

In this section we discuss the Russian language dataset for language modeling, its creation, and characteristics.

Penn Tree Bank (PTB) [13] is a very popular dataset for experimenting with language models, and many researchers use it in experiments with the same settings (dataset splits, etc.)—which allows to compare different language modeling approaches. PTB is heavily preprocessed, and the dataset vocabulary is limited to 10,000 tokens with no numbers, punctuation or capital letters.

Our goal was to create a similar reference dataset for the Russian language. Our dataset is based on the Lenta news dataset[2], a corpus of over 800K Russian news articles collected from Lenta.Ru between 1999–2019. To create our dataset we randomly sample sentences from the Lenta dataset after we apply preprocessing in a similar way as in PTB.

The *preprocessing pipeline* includes: a) lower-casing the text, b) replacing all URLs with a special token <url>, c) separating punctuation symbols from neighboring tokens with spaces, and finally d) replacing digits with a special character *D*, and for any 5 or more consecutive digits we use the special character *N*. Therefore, numbers in the dataset will take the following forms: *D, DD, DDD, DDDD, or N*. Step d) helps to keep meaningful numeric tokens, such as in these examples:

- часть D статьи DDD ук
- встреча так и завершилась со счетом D : D
- в DDDD году, госкорпорация обеспечивает DD процента электроэнергии .

Subsequently, we apply the NLTK[3] PunktSentenceTokenizer[4] to tokenize the text into sentences. We create a dictionary using the 15,000 most frequent tokens, and replace all other tokens in the text with <UNK>. In contrast to PTB, which uses a dictionary size of 10,000, we decided to keep 15,000 tokens because of the rich morphology of the Russian language where nouns, adjectives, and verbs change forms according to their role in the sentence.

Finally, we create the Russian language corpus by sampling 200,000 examples for the training set, 16,000 for the development set and 16,000 for the test set. In the sampling process, we only accepted sentences which fulfill the following conditions:

- The sentence contains less than 40 tokens.
- The sentence does not include any English words.
- The number of single (') and double (") quotation marks is even (balanced).
- Every opening bracket is followed by a closing bracket (balancing condition).
- Less than 10% of the tokens in the sampled sentence are the <UNK> token.

The dataset and code are available on GitHub[5]. Finally, we provide an overview of dataset statistics, including the mean number and standard deviation of the number

[2] https://github.com/yutkin/Lenta.Ru-News-Dataset

[3] https://nltk.org

[4] https://github.com/Mottl/ru_punkt

[5] https://github.com/zeinsh/lenta_short_sentences

of tokens per sentence in Table 1. The histogram and the box plot in Fig. 1 show the distribution of sentence length (number of tokens) in the dataset. Those statistics will be helpful to compare the sentences generated with various methods (see next section) with the original dataset.

Table 1: An overview of the Russian language modeling corpus (statistics of training, development and test set)

	Training set	Development set	Test set
#examples	200,000	16,000	16,000
Mean #tokens per example	13.26	13.17	13.14
Stddev for #tokens per example	5.63	5.66	5.62
#unique tokens	14,511	13,398	13,401

6. Experiments

6.1. Evaluation Setup

We experiment with the following popular text generation methods: VAE, GAN, and RNN.

For the **VAE** method, we build on the implementation by Baumgärtner[6] and add the implementation of additional scheduling methods, namely cyclical, constant and zero scheduling. Constant and cyclical scheduling are explained in Section 3, and in the zero schedule we set $\beta = 0$, which excludes the KL-divergence term from the lower-bound computation in equation (1). For the VAE encoder, we use 300-dim input embeddings, a single layer of 256 LSTM cells, and a latent vector (bottleneck) size of 16, and in the decoder again 256 LSTM cells and a 300-dim output. We train the VAEs for 10 iterations with an embedding dropout probability of 0.5.

The experiments with **seqGAN** are based on the PyTorch implementation on GitHub[7], We modified the original implementation to adapt it to our text dataset instead of discrete numbers created by the oracle generator in the original implementation. For seqGAN, we also use 300-dim input embeddings, and a single layer of 256 LSTM cells. We pretrain G for 10 iterations with an embedding dropout probability of 0.5, and then pretrain D (10 iterations). Then follow 10 epochs of adversarial training, each of which trains G for one iteration and D for five iterations. Finally, to measure the effect of the adversarial training on the RNNLM, we performed an evaluation on the pre-trained LSTM generator separately as **RNNLM** (see Section 6.2).

For evaluation, we use a dual strategy. First, in line with most research on language modeling, we calculate the *perplexity* on the test set for each model. Perplexity is a measure of how well a probability distribution predicts a sample. Perplexity does

[6] https://github.com/timbmg/Sentence-VAE

[7] https://github.com/suragnair/seqGAN

not always correlate with human correlation, in fact there is sometimes a negative correlation [2], for this reason we also include an human expert evaluation (see below).

Furthermore, we generate 10,000 sample texts with each method by greedily sampling word by word. Examples of the results are given in GitHub repository[8] and in **Appendix 1**. For the generated samples we use expert evaluation, where a Russian native speaker checks the generated text for grammatical correctness, and assigns a score of either 1 or 0. The value 1 signifies that no grammatical mistakes were found in the text. As we manually evaluate 100 unique sentences for each model, the maximum score per model is 100.

6.2. Evaluation Result

Here we analyze the models with regards to the following aspects: perplexity, token statistics of generated text, and manual evaluation of grammatical correctness. Furthermore, for VAE models we discuss the spatial distribution of latent representations.

In **Table 2** we report on the perplexity metric for the test set. The *zero* VAE model clearly shows the best results, followed by the cyclical model. As we will see in **Tables 3** and **4**, despite good results on perplexity, the *zero* model does not excel in grammatical correctness of the generated text.

Table 2: Perplexity calculated on test set for each model

x	zero	constant	linear	cyclical	RNNLM	seqGAN
perplexity	7.19	16.27	14.36	14.11	27.88	27.93

Table 3 gives an overview of some statistics of the generated text samples. We can see that LSTM RNNLM, seqGAN, and especially the *zero* variant of VAE produce a large number of unique sentences—with very little overlap with the training sentences. On the other hand, the *constant* VAE model fails to generate a large variety of sentences. Liu et al. [10] argue that the constant schedule ignores z and treats it as noise. Regarding sentence length, most models are similar, except *constant*, which creates shorter sentences. More interestingly, the *zero* and *constant* model show little variance in sentence length, while the other models much better capture the variance in sentence length of the training dataset. Finally, only RNNLM and seqGAN are able to have a diversity in vocabulary similar to the training data.

As mentioned, a Russian native speaker manually verified 100 generated sentences for each model regarding grammar. The results (number of grammatically correct sentences) are presented in **Table 4**. The data clearly shows that the *cyclical* VAE model performs very well (91% correct sentences), while the *zero* VAE model, although providing low perplexity, produces many grammatically wrong sentences. This corresponds with Chang et al. [2], ie. that perplexity does not always correlate with human evaluation. In our experiments, RNNLM and seqGAN fail to generate a high ratio of grammatically correct sentences.

[8] https://github.com/zeinsh/lenta_short_sentences/blob/master/samples.md

Table 3: Comparison between models regarding uniqueness and sentence length of the generated data

	zero	constant	linear	cyclical	RNNLM	seqGAN
# unique sent. (out of 10000)	10,000	231	8,810	8,868	9,972	9,979
# unique sent. not in train-set	10,000	225	8,694	8,752	9,972	9,979
Mean #tokens per sample	10.52	8.60	11.89	11.25	13.53	13.89
Std-dev of # tokens per sample	2.25	2.27	5.08	4.78	5.99	6.08
# unique words	5,324	353	5,028	4,550	11,430	11,505

Table 4: Number of grammatically correct sentences (out of 100) checked by a native speaker

	zero	constant	linear	cyclical	RNNLM	seqGAN
Score	77	79	86	91	43	51

Fig. 2 shows a projection of the latent representations in the development set into 2D space (using tSNE[9]). The figure compares the resulting distributions of the VAE methods with a zero, constant, linear and cyclical schedule. The figure shows that the zero schedule, which corresponds to a standard autoencoder, produces a rather irregular distribution of latent codes. The KL-divergence term in VAEs causes the algorithm to fill the latent space.

As discussed before, although zero gives the best perplexity on the test set, samples from the zero model contain many grammatically incorrect sentences, as it sometimes samples z from regions in the latent space with low density. That explains why samples from linear and cyclical VAES are better in terms of grammar, where latent codes produced by these models fill the latent space.

Finally, in the appendix and in the GitHub repository[10], we give examples on how VAE models can interpolate between two sentences. Following Bowman et al. [1], we sample two random points in the latent space and then decode those into two sentences. Then, starting from the first sentence, we gradually move through the latent space on a line to the second sentence, and pick points on the way, which are decoded into sentences. This process makes the ability of interpolation in the latent space explicit.

In summary, we found that the cyclical VAE produces best result with regards to grammatical correctness, followed by the VAE with linear schedule. Both also generate sentences with similar characteristics as in the training set (regarding sentence

[9] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

[10] https://github.com/zeinsh/lenta_short_sentences/blob/master/interpolation.md

length), however concerning sentence length and diversity of the used vocabulary the plain RNNLM and the seqGAN produce better results. But RNNLM and seqGAN generate a high number of grammatically wrong sentences in the experiments.

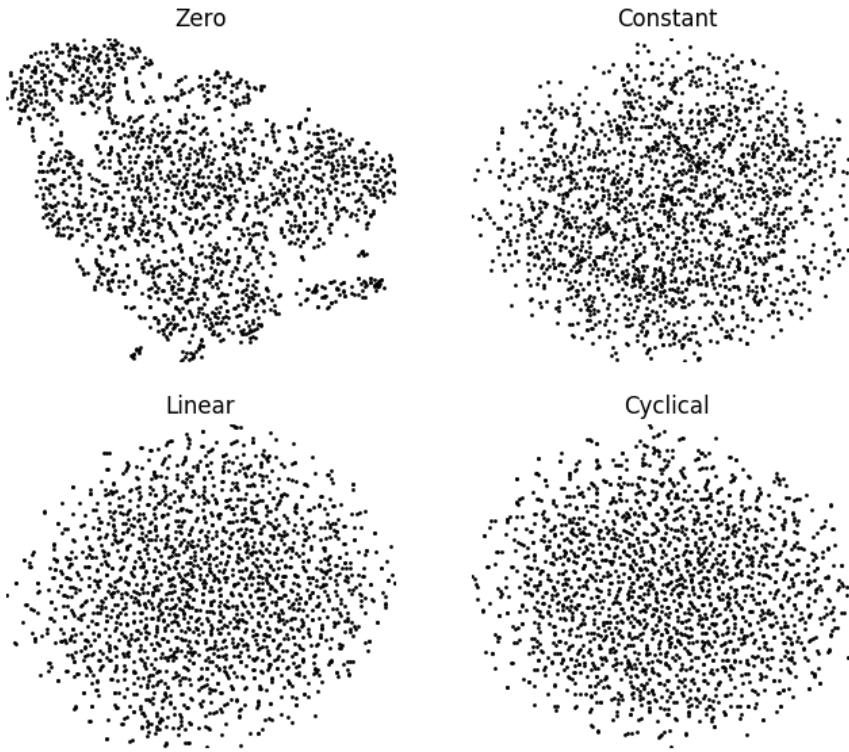

Figure 2: Latent representation of texts in the development set. The KL-divergence term in the constant, linear, and cyclical schedule forces the encoder to fill the latent space.

7. Conclusion

In this work, we present a new dataset for Russian language modeling (based on the Lenta News dataset) and perform a comparative study between two modern methods in text generation, namely VAE and seqGAN. Our results show the effect of the scheduling method on the quality of the generated text in VAEs, where linear and cyclical schedules produced the best models grammatically, however, the zero method showed the best perplexity, but an irregular distribution of the latent codes. LSTM and SeqGAN were able to replicate the mean and variance of the length of sentences in the original dataset as well as the number of unique words. The contributions of this work are: i) the provision (on GitHub) of a reference dataset for Russian

language modeling with 236K sentences in total, ii) the adaption of various VAE variants and seqGAN to Russian text, iii) and extensive experiments and evaluations with the chosen deep learning methods which indicate that the cyclical VAE approach performs best overall. Future work will include a deeper investigation of the latent representations produced by VAEs (and why VAEs produce less diverse sentences), apply state-of-the-art models like LeakGAN and studying the generation of Russian language text conditioned on topic, sentiment, etc.

8. Acknowledgments

This work was supported by the Government of the Russian Federation (Grant 074-U01) through the ITMO Fellowship and Professorship Program.

Appendix 1

In the appendix, we show the interpolation using the four VAE models trained on the reference dataset. As described in **Section 6.2** we sample two random points in the latent space and then decode those into two sentences. Starting from the first sentence, we gradually move through the latent space on a line to the second sentence, and pick points on the way, which are decoded into sentences. This processes makes the ability of interpolation in the latent space explicit.

VAE/zero schedule

- пожары с ними возник конфликт между двумя группами и на юго—востоке <unk> . <eos>
- пожары с ними возник конфликт между двумя группами и на юго—востоке <unk> . <eos>
- пожары с рельсов сошел с одной из самых опасных технологий , вызванных <unk> на работу . <eos>
- пожары с созданием по атомной энергии (магатэ) , на <unk> островах . <eos>
- пожары с созданием по спасению ракет у ворот у берегов острова , <unk> за рубеж . <eos>
- соперник по многим показателям проходит у дома от продажи билетов , <unk> за рубеж . <eos>
- соперник по многим показателям (нсн) создает у нее , <unk> за рубеж . <eos>
- соперник (по словам представителя рфс) при этом <unk> , сообщает тасс . <eos>
- соперник (по словам) у него возникли трудности , <unk> за собой . <eos>
- соперник—гран—при россии по легкой атлетике , <unk> за рубеж . <eos>

VAE/constant schedule

Explanation: This model decoded both points from the latent space into the *same* sentence.

- в dddd году он был объявлен в международный розыск . <eos>
- в dddd году он был объявлен в международный розыск . <eos>
- в dddd году он был объявлен в международный розыск . <eos>
- в dddd году он был объявлен в международный розыск . <eos>
- в dddd году он был объявлен в международный розыск . <eos>
- в dddd году он был объявлен в международный розыск . <eos>
- в dddd году он был объявлен в международный розыск . <eos>
- в dddd году он был объявлен в международный розыск . <eos>
- в dddd году он был объявлен в международный розыск . <eos>
- в dddd году он был объявлен в международный розыск . <eos>

VAE/linear schedule

- точная дата выхода фильма пока неизвестна , пока неясно , выйдет на экраны . <eos>
- точная дата выхода фильма пока неизвестна , пока неясно , не уточняется . <eos>
- такое заявление сделал на заседании совета федерации хоккея россии , передает риа новости . <eos>
- соответствующее заявление сделал на заседании совета федерации хоккея россии по футболу , передает риа новости . <eos>
- соответствующее заявление сделал на заседании совета федерации по правам человека , передает риа новости . <eos>
- соответствующее заявление сделал на заседании совета федерации по правам человека , передает риа новости . <eos>
- соответствующее заявление сделал в четверг , dd мая , на сайте следственного комитета рф . <eos>
- соответствующее заявление сделал в четверг , dd мая , в ходе совещания . <eos>
- соответствующее заявление в четверг , dd мая , приводит риа новости . <eos>
- соответствующее постановление опубликовано на сайте ведомства в четверг , dd мая , в <unk> . <eos>

VAE/cyclical schedule

- в то же время , по словам <unk> , он посетит россию , а также в последние годы . <eos>
- в то же время , по словам <unk> , он посетит россию , а также в последние годы . <eos>
- в то же время , по словам <unk> , он был вынужден уйти в отставку . <eos>
- в <unk> , где он жил в нью—йорке , не уточняется . <eos>
- в <unk> , где он жил в нью—йорке , не уточняется . <eos>

- в результате <unk> погибли dd человек , большинство из которых были убиты . <eos>
- по его словам , <unk> был задержан в ходе проверки , проведенной полицией . <eos>
- по его словам , <unk> был задержан в ходе перестрелки с полицией . <eos>
- по предварительным данным , <unk> был ранен , а также ранен . <eos>
- причины катастрофы устанавливаются , <unk> в результате инцидента никто не пострадал . <eos>

Appendix 2

Here we present a few examples of grammatically correct and incorrect sentences generated by the VAE/Cyclical model.

Grammatically correct

- если вина будет доказана , <unk> грозит до dd лет лишения свободы . <eos>
- но это не первый случай , когда он будет <unk> в прямом эфире . <eos>
- в последние годы он жил в нью—йорке и вашингтоне . <eos>
- стоимость контракта оценивается в dd миллионов долларов . <eos>

Grammatically incorrect

- если бы не удастся , то <unk> , как , будет <unk> , то , как он , не будет делать какие—то проблемы , касающиеся <unk> изменений в закон " о <unk> " . <eos>
- поединок состоялся в ночь на *D* февраля , однако известно , что <unk> в нем принял участие около dd тысяч человек . <eos>

References

1. *Bowman, S. R. et al.:* Generating sentences from a continuous space. arXiv preprint arXiv:1511.06349. (2015).
2. *Chang, J. et al.:* Reading tea leaves: How humans interpret topic models. In: Advances in neural information processing systems. pp. 288–296 (2009).
3. *Fedus, W. et al.:* Maskgan: Better text generation via filling in the _. arXiv preprint arXiv:1801.07736. (2018).
4. *Goodfellow, I. et al.:* Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014).
5. *Guo, J. et al.:* Long text generation via adversarial training with leaked information. In: Thirty-second aaai conference on artificial intelligence. (2018).
6. *Kingma, D. P., Welling, M.:* Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114. (2013).

7. *Kipyatkova, I. S., Karpov, A. A.:* A study of neural network russian language models for automatic continuous speech recognition systems. Automation and Remote Control. 78, 5, 858–867 (2017).
8. *Kuratov, Y., Arkhipov, M.:* Adaptation of deep bidirectional multilingual transformers for russian language. arXiv preprint arXiv:1905.07213. (2019).
9. *Kusner, M. J., Hernández-Lobato, J. M.:* Gans for sequences of discrete elements with the gumbel-softmax distribution. arXiv preprint arXiv:1611.04051. (2016).
10. *Liu, X. et al.:* Cyclical annealing schedule: A simple approach to mitigating kl vanishing. arXiv preprint arXiv:1903.10145. (2019).
11. *Merity, S. et al.:* Regularizing and optimizing LSTM language models. arXiv preprint arXiv:1708.02182. (2017).
12. *Merity, S. et al.:* Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843. (2016).
13. *Mikolov, T. et al.:* Empirical evaluation and combination of advanced language modeling techniques. In: 12th annual conf. of the international speech communication association. (2011).
14. *Mikolov, T. et al.:* Extensions of recurrent neural network language model. In: 2011 ieee intern. Conf. on acoustics, speech and signal proc. (ICASSP). pp. 5528–5531 IEEE (2011).
15. *Nesterenko, L.:* Building a system for stock news generation in russian. In: Proc. Of 2nd int. Workshop on nlg and the semantic web (webnlg 2016). pp. 37–40 (2016).
16. *Perera, R., Nand, P.:* Recent advances in natural language generation: A survey and classification of the empirical literature. Computing and Informatics. 36, 1, 1–32 (2017).
17. *Radford, A. et al.:* Language models are u/nsupervised multitask learners. OpenAI Blog. 1, 8, 9 (2019).
18. *Ruder, S. et al.:* Transfer learning in natural language processing. In: Proc. Of 2019 conf. Of the north american chapter of the association for computational linguistics: Tutorials. pp. 15–18 (2019).
19. *Sak, H. et al.:* Long short-term memory recurrent neural network architectures for large scale acoustic modeling. research.google. (2014).
20. *Shimorina, A. et al.:* Creating a corpus for Russian data-to-text generation using neural machine translation and post-editing. In: Proceedings of the 7th workshop on balto-slavic natural language processing. pp. 44–49 Association for Computational Linguistics, Florence, Italy (2019).
21. *Yang, Z. et al.:* Improved variational autoencoders for text modeling using dilated convolutions. In: Proc. 34th int. Conf. On machine learning-volume 70. pp. 3881–3890 JMLR. org (2017).
22. *Yu, L. et al.:* Seqgan: Sequence generative adversarial nets with policy gradient. In: 31st aaai conf. On art. Intelligence. (2017).
23. *Zhang, Y. et al.:* Adversarial feature matching for text generation. In: Proc. 34th int. Conference on machine learning-volume 70. pp. 4006–4015 JMLR.org (2017).