Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference “Dialogue 2023”

June 14–16, 2023

MaxProb: Controllable Story Generation from Storyline

Sergey Vychegzhanin
Vyatka State University
Kirov, Russia
vychegzhaninsv@gmail.com

Anastasia Kotelnikova
Vyatka State University
Kirov, Russia
kotelnikova.av@gmail.com

Alexander Sergeev
Vyatka State University
Kirov, Russia
sergeev.alexander0@gmail.com

Evgeny Kotelnikov
Vyatka State University
Kirov, Russia
kotelnikov.ev@gmail.com

Abstract

Controllable story generation towards keywords or key phrases is one of the purposes of using language models. Recent work has shown that various decoding strategies prove to be effective in achieving a high level of language control. Such strategies require less computational resources compared to approaches based on fine-tuning pre-trained language models. The paper proposes and investigates the method *MaxProb* of controllable story generation in Russian, which works at the decoding stage in the process of text generation. The method uses a generative language model to estimate the probability of its tokens in order to shift the content of the text towards the guide phrase. The idea of the method is to generate a set of different small sequences of tokens from the language model vocabulary, estimate the probability of following the guide phrase after each sequence, and choose the most probable sequence. The method allows evaluating the consistency of the token sequence for the transition from the prompt to the guide phrase. The study was carried out using the Russian-language corpus of stories with extracted events that make up the plot of the story. Experiments have shown the effectiveness of the proposed method for automatically creating stories from a set of plot phrases.

Keywords: text generation; decoding strategy; GPT
DOI: 10.28995/2075-7182-2023-22-539-553

MaxProb: Управляемая генерация историй на основе сюжетных линий

Вычегжанин С. В.
Вятский государственный университет
Киров, Россия
vychegzhaninsv@gmail.com

Котельникова А. В.
Вятский государственный университет
Киров, Россия
kotelnikova.av@gmail.com

Сергеев А. В.
Вятский государственный университет
Киров, Россия
sergeev.alexander0@gmail.com

Котельников Е. В.
Вятский государственный университет
Киров, Россия
kotelnikov.ev@gmail.com

Аннотация

Управляемая генерация историй по направлению к ключевым словам или выражениям является одной из целей использования языковых моделей. Недавние работы показали, что использование различных стратегий декодирования является эффективным подходом для достижения высокого уровня управления языком. Такие стратегии требуют меньше вычислительных ресурсов по сравнению с подходами, основанными на тонкой настройке предварительно обученных языковых моделей. В статье предложен и исследован метод управляемой генерации историй на русском языке *MaxProb*, работающий на этапе декодирования в процессе генерации

текста. Метод основан на использовании генеративной языковой модели для оценки вероятности ее токенов с целью смещения содержания текста к направляющему выражению. Идея метода заключается в генерации множества различных небольших по длине последовательностей токенов из словаря языковой модели, оценке вероятности следования направляющей фразы после каждой последовательности, и выборе наиболее вероятной последовательности. Метод позволяет оценить логичность последовательности токенов для перехода от затравки к направляющему выражению. Исследование проводились с использованием русскоязычного корпуса историй с выделенными событиями, составляющими сюжет истории. Эксперименты показали эффективность предлагаемого метода для автоматического создания историй из набора сюжетных фраз.

Ключевые слова: генерация текстов; стратегия декодирования; GPT

1 Introduction

Natural language generation (NLG) is one of the important areas of computational linguistics. It aims to produce plausible and readable text in a human language. In recent years, the use of large-scale pre-trained language models (PLMs), in particular transformer-based PLMs [21], has shown promising results, allowing generating more diverse and fluent texts. Modern neural network models such as GPT-3 [2] can create texts that are difficult to distinguish from texts written by a human.

NLG technologies are crucial in many applications such as dialogue and question-answering systems, story generation, advertising, marketing, product and service reviews.

Controllable Text Generation is a problem actively explored in NLG. This is the task of generating texts that meet certain control constraints set by a human [16]. Sentiment, keywords, events, etc. can be considered as such constraints. For example, when generating a story, it is important to control the storyline and the ending.

There are two types of control over text generation models: soft and hard control. The aim of soft control is, e.g., to provide the desired sentiment or topic of the generated text. Hard control requires ensuring that the text contains explicit constraints, e.g., certain keywords. Figure 1 shows an example of hard controllable text generation, where the story is generated according to the keywords provided by the storyline and the order in which they appear [25].

Storyline	needed → money → computer → bought → happy
Generated story	John needed a computer for his birthday. He worked hard to earn money. John was able to buy his computer. He went to the store and bought a computer. John was happy with his new computer.

Figure 1: Example of controllable story generation with hard control

Many existing controllable generation methods [5], [8], [25] require the creation of training corpora and the implementation of a training procedure that is labor intensive and time consuming. This paper overcomes this problem by developing a plug-and-play method applicable to any large-scale PLM. Currently, there are not enough studies on the controllable text generation in Russian, so the proposed method is tested on Russian language models and text corpora.

The idea of the method is to generate a set of short sequences of words that provide a coherent transition from the prompt to the guide phrase, and then estimate the probability of following the guide phrase after each generated sequence and choose the most probable sequence. This method is plug-and-play, i.e. it can be used with any autoregressive model. The experiments carried out on generating stories from a set of events that make up the plot of a story prove the effectiveness of the proposed method for creating texts from a set of plot phrases.

The contribution of the paper is as follows:

- we offer *MaxProb* – a method of controllable text generation that generates stories in accordance with a user-specified sequence of guide phrases that make up the plot of the story;
- we apply the method to the Russian language;
- we form a text corpus containing stories with extracted storylines;
- we experiment with story generation to confirm the effectiveness of the proposed method.

2 Previous work

This section discusses the existing methods of controllable text generation that can be applied to the problem of story generation, which is of primary research interest. Automated story generation is the problem of mechanically selecting a sequence of events or actions that meet a set of criteria and can be told as a story [11]. Each story has a story world, interacting characters, and objects. The complexity of the story generation task is to generate a coherent and fluent story that is much longer than the user-specified prompt.

Controllable generation methods can be classified into three categories [26]: fine-tuning, retraining or refactoring, post-processing. Fine-tuning PLMs on a specialized data set is the main way to interact with models. Methods of this type fine-tune some or all of the model parameters to create texts that satisfy certain constraints. Early work on controllable story generation used convolutional and recurrent neural networks. Fan et al. [6] used a two-stage hierarchical approach. At the first stage, using the convolutional neural network, a premise, which determined the structure of the story, was generated. Then the premise was converted into a text passage using the seq2seq model. Yao et al. [25] used the RAKE algorithm [18] to build a storyline for each story from the corpus at the training stage using the most important words. After the storyline was generated, the seq2seq model converted it into text.

Reinforcement learning can be used for controllable story generation. For example, Tambwekar et al. [20] developed a reward-shaping technique that produces intermediate rewards at all different time-steps, which are then back-propagated into a language model in order to guide the generation of plot points towards a given goal.

Later, pre-trained language models based on the Transformer architecture began to be used for controllable generation. The prompt-based approach became widespread. Li and Liang [12] proposed a method called “prefix tuning” that freezes the parameters of the PLM and performs error backpropagation to optimize a small continuous task-specific vector called “prefix”. A similar P-tuning method [10] differs from prefix tuning in that it does not place a prompt with the “prefix” in the input, but constructs a suitable template composed of the continuous virtual token, which is obtained through gradient descent.

Retraining or refactoring involves changing the architecture of the language model or retraining a model from scratch. This approach is limited by the insufficient amount of labeled data and the high consumption of computing resources. One of the first models in this direction was CTRL [8]. The model was trained on a set of control codes. Zhang et al. [27] proposed POINTER, an insertion-based method for hard-constrained text generation, which involves preserving of specific words.

Cho et al. [4] proposed Story Control via Supervised Contrastive learning model to create a story conditioned on genre. The model learns conditional probability distribution by supervised contrastive objective, combined with log-likelihood objective.

Methods based only on using a decoder are called post-processing. Such methods require less computational resources. A representative of this group of methods is PPLM [Dathathri et al., 2020], which first trains an attribute discriminant model and then uses it to guide language model to generate the text with corresponding topic or sentiment. This group also includes the Keyword2Text method [15], which can be applied to an existing autoregressive language model without additional training. The idea of the method is to shift the output distribution of the language generation model to the semantic space of a given guide word in the word2vec or GloVe vector space. A similar idea is used in [22], but the difference is that the score function of the autoregressive language model is modified with the score function of another language model from the family of autoencoding models rather than with the cosine similarity to the target keyword.

Yang et al. [24] developed the Re3 framework to automatically generate longer stories of over two thousand words. Re3 first creates a structured plan, setting and characters by prompting GPT-3 with a premise. Then Re3 injects contextual information from both the plan and current story state into new GPT-3 prompt to generate new story passages.

In this paper, we propose a post-processing method that implements a decoding strategy based on heuristics. The difference from previous works [15], [22] lies in the fact that at each generation step for small sequences of tokens, the probability of following the guide phrase is estimated. The method is based on the idea that choosing a sequence of tokens, after which the probability of following the guide phrase is maximum, will induce the model to generate text, shifting its content to the guide phrase.

3 Controllable text generation

In this paper, we consider conditional probabilistic models for which the probability of the output text $X = \{x_1, \dots, x_n\}$ can be factorized by tokens:

$$P(X) = \prod_{i=1}^{n} P(x_i | x_{<i}), \quad (1)$$

where x_i denotes the i-th output token, and $x_{<i}$ denotes previous tokens $x_1, \dots, x_{i-1}$.

In accordance with formula (1), the goal of conditional text generation can be formulated as follows:

$$P(X|C) = \prod_{i=1}^{n} P(x_i | x_{<i}, C), \quad (2)$$

where C denotes the control conditions and X is the generated text, which complies with the control conditions.

While generating, sequences of natural language units (symbols, words, or sentences) are decoded from the probability distribution P. The decoding strategy plays an important role. At each time step, it selects tokens from the probability distribution over a model vocabulary. Beam search [14] and nucleus sampling [7] are examples of known decoding strategies.

Generative language models such as GPT learn to predict the next token in a given sequence of tokens. Text generation is a natural application for such models. However, when predicting the next token of a sequence, they are not able to take into account the context following it, which is supposed to be the content of the generated text.

In this study, we propose the *MaxProb* method, which at each generation step determines the most probable sequence of tokens for logically linking the prompt and the guide phrase that should be used in the text. The idea of the method is based on using intrinsic knowledge of a pre-trained language model to evaluate the token sequences and select the appropriate sequence for a coherent transition to the guide phrase. The proposed method can be applied to any autoregressive language model.

Let us consider the sequence $X = \{x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_{i+k}, t, \dots, t_m\}$. For a given prompt $X_{1:i-1} = \{x_1, \dots, x_{i-1}\}$ and a guide phrase $T = \{t_1, \dots, t_m\}$ theoretically it is possible to find the connecting sequence $X_{i:i+k} = \{x_i, x_{i+1}, \dots, x_{i+k}\}$ using exhaustive search of tokens from the model vocabulary. However, such search has an exponential dependence on the length of the connecting sequence and is not applicable in practice. Therefore, in order to reduce the number of variants we propose a heuristic technique for generating and evaluating connecting sequences (Fig. 2).

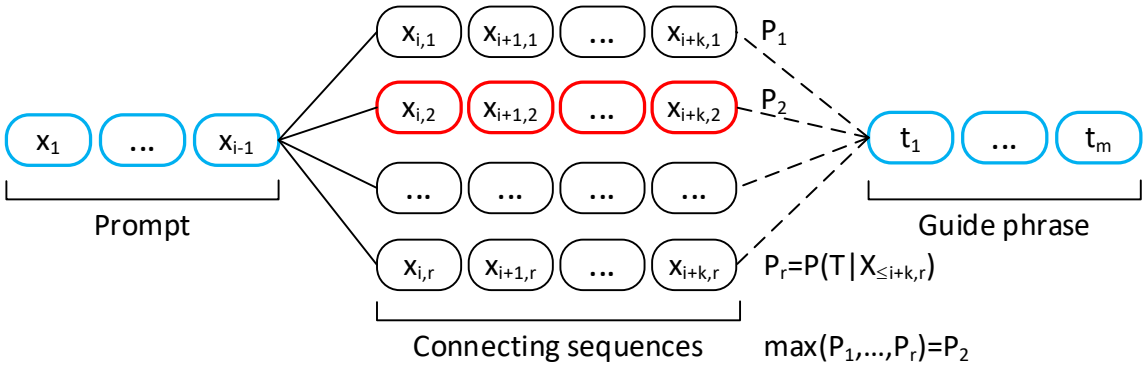

Figure 2: *MaxProb* method scheme

First, as continuations of the prompt $X_{1:i-1}$, r different sequences of tokens of length $k+1$ are generated using some decoding strategy. Next, for each of the r sequences, the probability of following the guide phrase T after it is determined by the formula:

$$P(X_{i:i+k}|X_{1:i-1}, T) = P(T|X_{\leq i+k}) = \prod_{j=1}^{m} P\left(t_j | t_{<j}, X_{\leq i+k}\right). \quad (3)$$

Further, at the current generation step, a sequence is selected for which the probability (3) is maximum, and the sequences of length $k + 1$ are repeatedly generated. In order to fulfill the condition of the explicit presence of the guide phrase in the text, after the generation of a given number of tokens is completed, this phrase can be inserted in the position in the text where it had the maximum probability for the entire generation time. After the phrase is inserted, the generation can continue towards the next guide phrase.

Formula (3) makes it possible to estimate the probability of following the guiding phrase for each connecting sequence of tokens, but does not evaluate their semantic similarity. There may be cases where semantic similarity is more important than the likelihood of following the guide phrase. To assess the similarity of the connecting sequence and the guide phrase, it is proposed to use the Jaccard coefficient:

$$K_J = \frac{C}{A + B - C}, \tag{4}$$

where A is the set of words in normal form from the prompt, B is the set of words in normal form from the guide phrase, C is the set of common words for the prompt and the guide phrase.

Taking into account formulas (3) and (4) for connecting sequences, the average score, which establishes a balance between the two measures, can be determined by the formula:

$$Score_{x_{i:i+k}} = w_{prob} P_{norm} + w_J K_J, \tag{5}$$

where w_{prob}, w_J are weight coefficients, P_{norm} is the normalized probability of following the guide phrase.

Thus, at each time step, the proposed method allows selecting the most logical sequence of tokens for linking the prompt and the guide phrase, based on the knowledge of the generative model itself.

As an example of how the method works, let us consider a text at some *i*-th generation step and a guide phrase separated by a sequence of unknown tokens, for example, of length 3 (Fig. 3). In the figure, the prompt for the autoregressive model is highlighted in blue, and the guide phrase is highlighted in orange. The connecting sequence is marked with labels <x₁><x₂><x₃>.

Однажды в лесу, около речки, сидел мальчик с бабушкой. Вдруг в это время из-за **<x₁><x₂><x₃>** волк напал на ребенка

Once in the forest, near the river, a boy was sitting with his grandmother. Suddenly, at this time, **<x₁><x₂><x₃>** the wolf attacked the child

Score	*P*	K_J	<x₁><x₂><x₃>, Russian	<x₁><x₂><x₃>, English
0.944	3.20E-11	0.200	кустов вышли волки	wolves came out from behind the bushes
0.226	6.50E-12	0.200	поворота вышел волк,	a wolf came out from around the corner,
0.105	1.90E-13	0.100	дерева на поляну	from behind a tree to a clearing
0.100	4.60E-18	0.100	дерева выскочило	from behind a tree jumped out
0.100	9.30E-19	0.100	деревьев вышел лев,	a lion came out from behind the trees,
0.100	5.70E-22	0.100	деревьев вышли три	from behind a tree appeared three
0.100	3.00E-24	0.100	деревьев показалась большая	from behind a tree appeared a large
0.052	2.80E-13	0.100	деревьев выскочили	from behind the trees jumped out
0.048	1.70E-13	0.100	поворота леса вышел	from around the corner of the forest came out
0.044	9.40E-15	0.100	поворота речки выскочил	out of the turn of the river, jumped out

Figure 3: Example of prompt and connecting sequences at the *i*-th generation step

The prompt is an input of the autoregressive model. With some decoding strategy, such as top-k sampling, r different sequences of 3 tokens <x_1><x_2><x_3> are generated. For them, the probabilities of following the guide phrase P and the Jaccard coefficients K_J are calculated. The calculated values are averaged by formula (5). The sequences of tokens are sorted in descending order of *Score*, and the sequence with the highest value of the average score is selected. The selected sequence is attached to the prompt, and the generation process continues until the specified number of tokens is generated.

4 Text corpus

To conduct experiments, a text corpus[1] was formed from fairy tales in Russian with extracted storylines. The corpus is made up of fairy tales placed on nukadeti.ru[2] with a length of no more than 5000 characters. In total, the training corpus contains 562 fairy tales.

In each fairy tale, plot phrases were singled out, i.e. phrases that determine the main events in the story, the storyline. To do this, first, in each fairy tale keywords and phrases were selected, using the methods *yake*[3] [3], *rakun*[4], *frake*[5], *textrank*[6], *rutermextract*[7], *keybert*[8] methods. Each method selected 15 keywords and phrases. The *yake* and *rutermextract* methods showed the best quality, so their results were used in the next stage to compose plot phrases.

The *yake* and *rutermextract* methods were selected out of six methods manually. The main problems with other methods were the following. The top keywords and phrases of the *rakun* and the *keybert* were very often parts of each other, they intersected, i.e. were parts of one longer phrase. So, the number of sentences with these selected keywords was very low and the plot could not be built out of them.

The *frake*'s results often contained just single words and it was very difficult to understand from which sentences they were selected (if they repeated several times).

The problem of *textrank* was that it didn't pay attention to sentence segmentation – many selected phrases were parts of two neighbor sentences.

Further, plot phrases were extracted from fairy tales according to the following algorithm:

1. Events were found. Events are syntactically related triples <object, action, object> (for example, "старуха, испекла, колобок" – "old woman, baked, bun"). The objects were selected from a set of keywords, and the actions was determined from the parse tree as nodes, syntactically associated with the objects. The stanza library[9] was used to make the syntax parsing of the sentences.

2. The most important events found were selected from the found events. Each selected event was assigned a weight obtained by summing the weights of the keywords extracted by the *yake* and *rutermextract* methods separately.

3. From the selected important events, a plot phrase was formed, determined by a 4-element set (o_1, v, o_2, m), where v is a verb, o are objects related to the verb, m is a modifier, prepositional object, or indirect object. Prepositions are possible before o and m. An example of an event: "grooves in the forest spilled into whole streams", where "spilled" is v, "grooves" and "streams" are o, "forest" is m ("канавки в лесу разлились в целые ручьи", v – "разлились", o – "канавки", "целые ручьи", m – "лесу").

For each of the two methods for extracting keywords, their own plot phrases were formed, the number of which, depending on the fairy tale, varied from 0 to 26. Figure 4 shows the distribution of the number of plot phrases extracted using the *yake* and *rutermextract* methods.

[1] https://github.com/icecreamz/MaxProb.
[2] https://nukadeti.ru.
[3] https://github.com/LIAAD/yake.
[4] https://github.com/SkBlaz/rakun.
[5] https://github.com/cominsys/FRAKE.
[6] https://github.com/JRC1995/TextRank-Keyword-Extraction.
[7] https://github.com/igor-shevchenko/rutermextract.
[8] https://github.com/MaartenGr/KeyBERT.
[9] https://stanfordnlp.github.io/stanza.

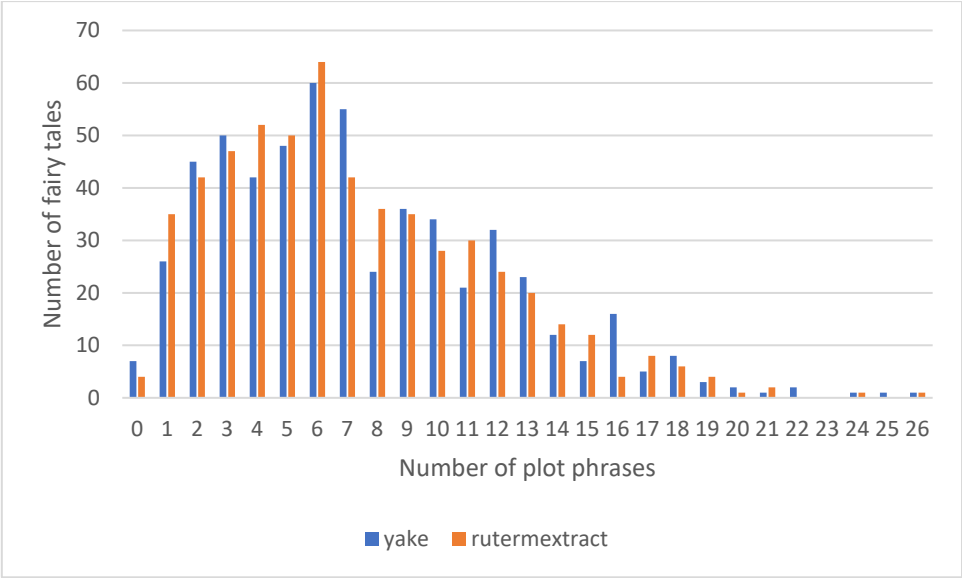

Figure 4: Distribution of the number of plot phrases

The number of sentences in fairy tales varied from 4 to 139. The distribution of the number of sentences is shown in Fig. 5.

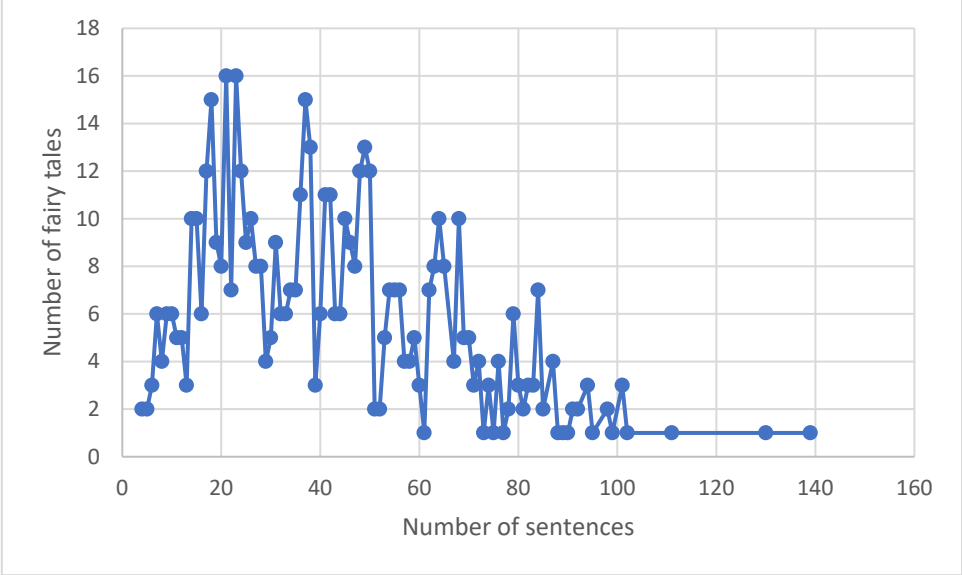

Figure 5: Number of sentences in fairy tales

Since the number of resulting plot phrases should correlate with the length of the tale, the plot was assembled from the selected phrases according to the following algorithm:

1. The minimum number of phrases in the plot is 1, the maximum is the rounded-up value of the logrhyme to base 2 of the number of sentences n in the text: $\lceil log_2 n \rceil$.
2. If the *yake* method returned the number of plot phrases in the above range, these phrases were taken in order as a plot.
3. If the *yake* method produced fewer plot phrases, and the *rutermextract* method yielded enough, then the *rutermextract* phrases were taken in order as a plot.
4. If both methods returned the number of phrases less than the minimum value, their results were combined without repetitions in the order of the sentences in the text.

5. If the *yake* method produced more plot phrases than the maximum allowable in accordance with point 1, then a part of the fragments with maximum weights was taken for the required amount.

Table 1 shows the distribution of the number of phrases in the plot in the training corpus. The first column contains the number of phrases in the plot, the second – the number of fairy tales with such a number of phrases, the third – the share of the total number of fairy tales in the training corpus, i.e., from 562 fairy tales.

A test corpus of 25 plots was also formed. The distribution by the number of plot phrases in the test corpus is proportional to the distribution in the training corpus and is given in the fourth column of Table 1.

# Plot phrases	# Fairy tales in the training corpus	Share of the total number of fairy tales, %	# Fairy tales in the test corpus
1	31	5.46	1
2	48	8.45	2
3	53	9.33	2
4	56	9.86	3
5	107	18.84	5
6	185	32.57	8
7	80	14.08	4
8	2	0.35	0

Table 1: Distribution of the number of phrases in the plot

Table 2 shows statistics on the number of tokens received using the ruGPT-3 Large tokenizer in fairy tales of training corpus, depending on the number of plot phrases.

# Plot phrases	Minimum number of tokens	Maximum number of tokens	Average number of tokens
1	28	900	230.9
2	85	400	238.9
3	115	1,015	344.3
4	128	752	308.9
5	212	950	476.4
6	406	1,283	796.0
7	757	1,503	1,150.1
8	1,555	1,897	1,726.0

Table 2: Number of tokens

5 Experimental Setup

Keywords used in plot events were extracted from texts using the *yake* and *rutemextract* libraries. The initial word forms for calculating the Jaccard coefficient were determined using the pymorphy2 library [9]. Text generation experiments were carried out using the ruGPT-3 Large[10] language model (760 million parameters), which is the Russian-language version of the GPT-2 model [17].

In the experiments, fairy tales were generated according to a given sequence of events that determines the plot of the fairy tale. The top-k sampling decoding strategy with parameter $k = 10$ was used as a decoding strategy in MaxProb to obtain connecting sequences of tokens.

The values of the weight coefficients in formula (5) were determined empirically based on the analysis of the generated connecting sequences. The coefficients took the values $w_{prob} = 0.9$ and $w_J = 0.1$. The probability of following the guide phrase turned out to be more significant, and due to the w_J coefficient, the connecting sequence that was closest in content to the guide phrase was ranked first.

[10] https://huggingface.co/sberbank-ai/rugpt3large_based_on_gpt2.

The length of connecting sequences was 3 tokens. Experiments were also carried out for windows ranging in size from 1 to 15 tokens. According to the results of the experiments, a small window of connecting sequences had a better effect on shifting the content of the generated text towards the plot phrase than a large window. With a large window size, suitable short sequences of words, most likely followed by a guide phrase, could be missed, and as a result, the content of the generated text deviated significantly from the content of the guide phrase.

The maximum length of the generated fairy tale (in tokens) depended on the number of plot phrases and was equal to the average number + 10% of the tokens (see Table 2).

The proposed method was compared with three methods of controllable text generation:

1. Inserting key phrases in a prompt (PromptLearn).

When conducting experiments using the PromptLearn method, the ruGPT-3 Large model was fine-tuned with 80% of the tales from the training corpus for three epochs. The prompt with size up to 1024 tokens was used as input data for the model:

"Plot: {plot phrase 1}, {plot phrase 2}, …, {plot phrase n}.\n
Text: {the text of fairy tale}"

For each tale, the number of plot phrases ranged from 1 to 8. To generate fairy tales, sampling was used with parameters $p = 0.95$ and $k = 50$. The length of the generated fairy tale was chosen similarly to MaxProb.

2. Few-shot learning (FewShotLearn).

The ruGPT-3 Large model was also used to apply the FewShotLearn method. The prompt was used as input for the model:

"Compose text with keywords:\n
Plot: {plot phrase 1}, {plot phrase 2}, …, {plot phrase n}.\n
Text: {the text of fairy tale} ###
Plot: {plot phrase 1}, {plot phrase 2}, …, {plot phrase n}.\n
Text: {the text of fairy tale}"

The number of fairy tales input to the model depended on the estimated maximum length of the generated text so that the total input sequence fit into 2048 tokens allowed for the model. The range of the number of input training examples is from 1 to 5, most often 3. When generating texts, the same parameters as for PromptLearn were used. The length of the generated fairy tale was chosen similarly to MaxProb.

3. Constrained beam search (ConstrainedBS).

ConstrainedBS was used as the baseline of controlled generation. Plot phrases were tokenized and used as a list of restrictions. The generation was carried out using the ruGPT-3 Large model. The prompt "Однажды" ("Once") was used as an input of the model. The number of beams varied from 7 to 10 to generate different stories. A prohibition on the repetition of 3-grams was also established. The length of the generated fairy tale was chosen similarly to MaxProb.

The quality of the generated texts was evaluated using automatic and human-centric evaluation methods. Four measures were used for automatic evaluation [13], [23], [28]:

- perplexity (PPL) – is a metric to measure how well the language probability model predicts a sample. It is usually calculated as the exponential mean of the negative log-probability per token in the language model. We calculated perplexity using the ruGPT-3 Medium[11] language model (350 million parameters);
- repetition (Rep) evaluates the proportion of repeated 4-grams in the text, where the tokens belong to the vocabulary of the ruGPT-3 Large model;
- Word Inclusion Coverage (Cov) shows the percentage of plot words included in the generated text. Plot and generated words are lemmatized;
- self-BLEU-5 evaluates the syntactic diversity of a given set of texts. It is defined as the average overlap between all generated texts.

[11] https://huggingface.co/sberbank-ai/rugpt3medium_based_on_gpt2.

Three measures were used for human-centric evaluation:
- coherence – whether the story is consistent in terms of causal relationships in the context;
- relevance – the story corresponds to the plot, the events in the story unfold in accordance with the storyline;
- interestingness – how the user likes the story, whether it is interesting.

6 Results and discussion

Table 3 shows the statistical characteristics of the generated texts, calculated using the GEM-metrics library[12]:
- Avg length – the average length of texts (in words);
- Vocab size – the number of different words;
- Distinct-n – the ratio of distinct n-grams over the total number of n-grams.

Generation methods	Avg length	Vocab size	Distinct-1	Distinct-2	Distinct-3
ConstrainedBS	447	3,149	0.11	0.49	0.85
FewShotLearn	158	1,998	0.19	0.57	0.77
PromptLearn	430	3,608	0.13	0.50	0.77
MaxProb	497	3,015	0.10	0.41	0.70

Table 3: Statistical characteristics of generated texts

Analyzing Table 3, you can see that the FewShotLearn method, on average, generated fairy tales 3 times shorter than the other three methods. It should be noted that when generating longer tales, the first tale was often interrupted and a new tale began.

Table 4 shows the average values of perplexity, repetition, word inclusion coverage, and self-BLEU-5 measures calculated for fairy tales generated from 25 storylines of test corpus. For each storyline, two fairy tales were generated. A total of 50 tales were generated by each method.

Additionally, the scores were also calculated for the base model ruGPT-3 Large. The ruGPT-3 Large model was preliminarily fine-tuned on the training corpus of fairy tales with the addition of the prefix "Текст: " ("Text: ") to the beginning of each fairy tale, which was then used as a prompt during generation. The experiments used the strategy of decoding top-k sampling with the parameter $k = 10$.

Generation methods	↓ PPL ± Std	↓ Rep, %	↑ Cov, %	↓ Self-BLEU-5
ruGPT-3	**5.3 ± 1.5**	26.43	20.07	0.028
ConstrainedBS	6.8 ± 2.5	**0.61**	80.86	0.094
FewShotLearn	9.9 ± 6.1	16.40	43.49	**0.014**
PromptLearn	6.8 ± 1.7	14.82	71.32	0.032
MaxProb	7.0 ± 1.4	18.33	**99.54**	0.063

Table 4: Automatic quality scores for generation methods

The values of the Cov measure in Table 4 show that the MaxProb method ensures that more than 99% of the words from the storyline events appear in the text. The texts generated by this method met the requirement of matching the storyline to the best extent. The smallest number of words from the storyline appeared in the texts generated by the FewShotLearn method and is 43.49%. In such texts, the required characters and events were rare. This is largely due to the relatively short length of the generated tales.

The values of the Rep measure for the FewShotLearn, PromptLearn, and MaxProb methods are quite close to each other and vary from 14.82% to 18.33%. The ConstrainedBS method has a Rep value close to zero as a result of setting the prohibition on the repetition of 3-grams, otherwise the generation was often reduced to repetitions of words. Repeatability values do not suggest a significant superiority of

[12] https://github.com/GEM-benchmark/GEM-metrics.

one method over others. Notably, controllable generation methods reduced the repeatability value compared to the ruGPT-3 base model.

The lowest PPL value among controllable generation methods was obtained for PromptLearn and ConstrainedBS and is 6.8. The MaxProb method showed a 0.2 higher average PPL, but it has a lower standard deviation, i.e. provides a more stable level of perplexity. For the FewShotLearn method, perplexity and standard deviation were the highest. It is known, that a lower perplexity value corresponds to a better model. The increase in perplexity compared to the base ruGPT-3 model indicates that the control process is “unnatural” for the model. This causes the model to be more "surprised" by the tokens observed in the text.

The self-BLEU-5 measure has the lowest value for FewShotLearn. The texts generated by this method turned out to be the most syntactically diverse. The variety of PromptLearn is at the level of the basic ruGPT-3 model. The least varied texts are for the ConstrainedBS method.

To calculate human-centric measures, the generated texts were evaluated by three annotators for coherence, relevance, and interestingness. The assessment was carried out on a 5-point Likert scale (1 – the worst, 5 – the best). For all the methods, only the generated sequence was evaluated, without prompt. Inter-annotator agreement was measured using the Spearman coefficient [1]. The value of this coefficient for the “coherence” criterion was 0.54, “relevance” – 0.87, “interestingness” – 0.59. The values, which are greater than 0.5 indicate high annotator agreement [19].

Table 5 shows the average scores of coherence, relevance and interestingness.

Generation methods	**↑ Coherence**	**↑ Relevance**	**↑ Interestingness**
ConstrainedBS	1.65	2.91	1.56
FewShotLearn	2.23	1.63	2.25
PromptLearn	**2.62**	2.23	**2.82**
MaxProb	2.20	**4.89**	2.74

Table 5: Human-centric quality scores for generation methods

The coherence scores for all methods turned out to be low, less than 3 points. The low coherence is due to the quality of the ruGPT-3 base model, which was used in the experiments. The PromptLearn method turned out to be the best in terms of coherence, the MaxProb method more often violated the coherence, and ConstrainedBS generated practically incoherent texts. However, MaxProb almost always ensured that all events from the storyline appeared in the text, as evidenced by a high relevance score. Despite the lowest coherence, the texts with MaxProb were slightly less interesting than with the PromptLearn method, but were more interesting than with FewShotLearn.

Figure 6 shows the parallel coordinates visualization of all calculated measures.

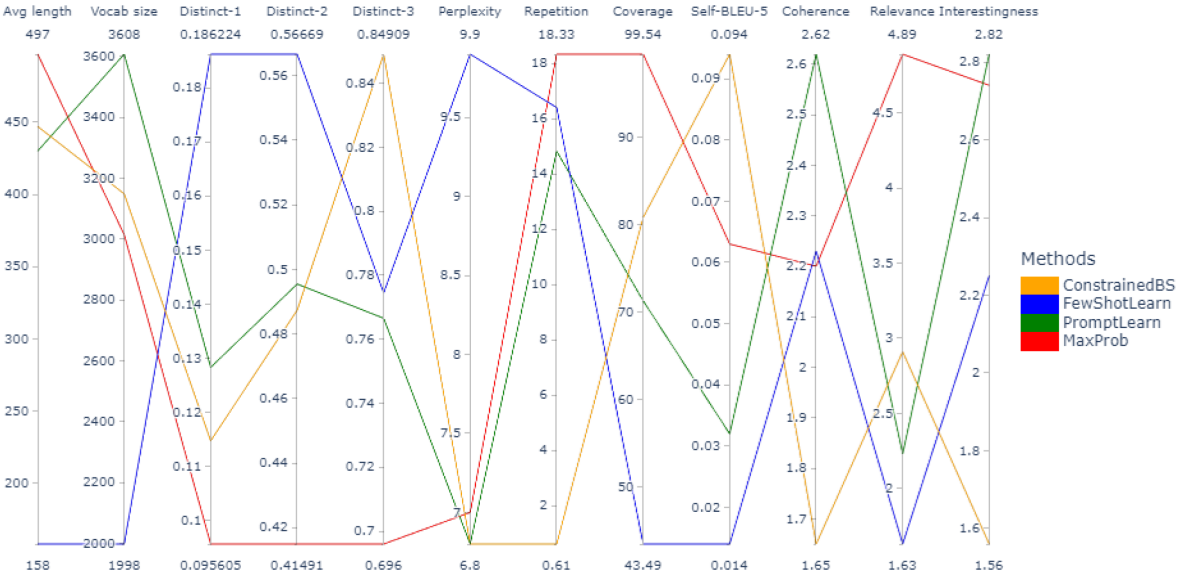

Figure 6: The parallel coordinates visualization of the measures

Let us give a specific example of the MaxProb method (Fig. 7). For the guide phrase “the cat ate sour cream” (“кот съел сметану”) for some *i*-th step, the text “An old woman had a cat, whom she loved very much and called: Ko-ko-ko. The cat loved” (“У одной старушки был кот, которого она очень любила и которого звала: Ко-ко-ко. Кот очень любил”). At the *i*-th step, using the decoding strategy top-*k* sampling, the connecting sequences of three tokens were obtained, shown in Fig. 7. For each sequence, the probabilities of following the guide phrase P by formula (3), the Jaccard coefficients K_J by formula (4) and the average values of *Score* by formula (5) are calculated. According to the results of the *i*-th step, the sequence “milk with bread,” (“молоко с хлебом,”) was chosen, which has the highest average *Score*.

У одной старушки был кот, которого она очень любила и которого звала: Ко-ко-ко. Кот очень любил **<x₁><x₂><x₃>** кот съел сметану

An old woman had a cat, whom she loved very much and called: Ko-ko-ko. The cat loved **<x₁><x₂><x₃>** the cat ate sour cream

Score	*P*	K_J	**<x₁><x₂><x₃>, Russian**	**<x₁><x₂><x₃>, English**
0.900	2.50E-10	0.111	молоко с хлебом,	milk with bread,
0.121	5.90E-12	0.143	старушку,	old woman,
0.113	3.60E-12	0.143	, чтобы его	, to be
0.105	1.30E-12	0.143	свою кошку и	his cat and
0.104	1.20E-12	0.143	ее, да	her, yes
0.102	6.60E-13	0.143	ее и не	her and not
0.101	1.60E-13	0.143	, когда его	, when he
0.100	6.40E-15	0.143	ее, Она	her, She
0.100	1.40E-15	0.143	эту старушку	this old woman
0.066	6.20E-12	0.125	молоко, и,	milk, and,

Figure 7: Connecting sequences and their scores on the *i*-th step of generation

Table 6 shows the connecting sequences for steps $i + 1$ through $i + 5$. The sequences that received the highest *Score* value are highlighted in blue at each step. These sequences were chosen as the most probable ones and added to the prompt.

№	Step $i + 1$	Step $i + 2$	Step $i + 3$	Step $i + 4$	Step $i + 5$
1	а еще больше	- сметану	. И вот	однажды кот съел	всю сметану
2	и, когда	любил сме- тану,	. А еще	однажды, когда	сметану и
3	а хлеб -	со сметаной	с молоком.	однажды вече- ром кошка	сметаны,
4	а больше всего	ел сметану,	, но молоко	однажды утром старушка	столько сме- таны,
5	поэтому, как	- с молоком,	, и поэтому	он как-то	все сметанное
6	но не любил,	с молоком,	, поэтому каж- дый	он любил сме- тану	все сметаны
7	и поэтому он	любил, когда	, которая была	он, чтобы	все молоко,
8	но он не	со сливочным	и хлеб.	, когда он	всё молоко,
9	и если молоко	с капустой,	. Поэтому ба- бушка	, однажды кот	всё, что
10	но молока в	с сыром,	. Кот ел	, как-то	целый хлеб и

№	Step $i+1$	Step $i+2$	Step $i+3$	Step $i+4$	Step $i+5$
1	and even more	– sour cream	. And then	one day the cat ate	all the sour cream
2	and, when	loved sour cream,	. And then	one day, when	sour cream and
3	and bread –	with sour cream	with milk.	Once in the evening the cat	sour cream,
4	and most of all	ate sour cream,	, but milk	Once in the evening the old woman	so much sour cream,
5	that's why, how	– with milk,	, and that's why	he once	all of sour cream
6	he didn't liked,	with milk,	, that's why every	he liked sour cream	all sour cream
7	and that's why he	liked, when	, which was	he, to	all milk,
8	but he didn't	with creamy	and bread.	, when he	all milk,
9	and if milk	with cabbage,	. That's why the old woman	, once the cat	all, that
10	but milk in	with cheese,	. The cat ate	, once	whole bread and

Table 6: Connecting sequences on steps $i+1$, …, $i+5$ of generation: Russian (top) and English (bottom) versions

As a result, after $i+5$ steps, the text was generated: "An old woman had a cat, whom she loved very much and called: Ko-ko-ko. The cat loved milk with bread, and even more – sour cream. And then one day the cat ate all the sour cream". This example demonstrates that choosing a sequence after which the probability of a guide phrase is maximum induces the generative model to lead the text to the required phrase. At the same time, the connecting sequence may not contain the guide phrase in an explicit form, but be close to it in meaning due to synonyms.

7 Conclusion

The proposed *MaxProb* method allows generating stories in accordance with a user-specified sequence of guide phrases that determines the plot of the story. Guide phrases describe some of the key events in the story and consist of several words. The method uses a generative language model to estimate the probability of following a guide phrase after various short sequences of tokens generated by the model. The method selects the sequence with the highest probability, prompting the model to shift the content of the text towards the guide phrase. Experiments carried out using the Russian-language corpus of fairy tales with extracted storylines showed that the proposed method provides a high proportion of story words (more than 99% in Cov) and phrases (4.89 points in Relevance) in the text. In terms of text quality (PPL measure and interestingness), the method is comparable to the PromptLearn fine-tuning method, but it does not require creating a training corpus and the executing of a time-consuming training procedure.

Ethical considerations

The proposed method helps to control the content of automatically generated text according to the user's needs. Note that large language models, including the one used in the proposed ruGPT-3 method, generate texts similar to texts written by a person. However, it is not guaranteed that the generated texts are factually correct. They may contain false or fictitious information that may mislead the non-expert reader. When using plot phrases containing factually incorrect information, the generation will be based on false content and, therefore, will lead to the creation of inaccurate texts. Like any tool, it can be used for negative purposes. Content control can lead to the creation of fake text for the purpose of deception, disinformation or propaganda. We hope that our method will be used for positive purposes, like helping writers to create fairy tales in accordance with a given plot. Placing such methods in the public domain will help develop countermeasures to detect them.

Acknowledgements

This work was supported by Russian Science Foundation, project № 23-21-00330, https://rscf.ru/en/project/23-21-00330/.

References

[1] Amidei J., Piwek P., Willis A. Agreement is overrated: A plea for correlation to assess human evaluation reliability // Proceedings of the 12th International Conference on Natural Language Generation. – 2019. – P. 344–354.
[2] Brown T.B., Mann B., Ryder N., Subbiah M., Kaplan J. et al. Language models are few-shot learners // Advances in Neural Information Processing Systems. – 2020. – Vol. 33. – P. 1877–1901.
[3] Campos R., Mangaravite V., Pasquali A., Jorge A., Nunes C., Jatowt A. YAKE! Keyword Extraction from Single Documents using Multiple Local Features // Information Sciences Journal. – 2020. – Vol. 509. – P. 257–289.
[4] Cho J., Jeong M., Bak J., Cheong Y.-G. Genre-controllable story generation via supervised contrastive learning // Proceedings of the ACM Web Conference 2022. – 2022. – P. 2839–2849.
[5] Dathathri S., Madotto A., Lan J., Hung J., Frank E., Molino P., Yosinski J., Liu R. Plug and play language models: A simple approach to controlled text generation // Computing Research Repository. – 2020. – arXiv:1912.02164. – Access mode: https://arxiv.org/abs/1912.02164.
[6] Fan A., Lewis M., Dauphin Y. Hierarchical neural story generation // Computing Research Repository. – 2018. – arXiv:1805.04833. – Access mode: https://arxiv.org/abs/1805.04833.
[7] Holtzman A., Buys J., Du L., Forbes M., Choi Y. The curious case of neural text degeneration // Proceedings of the 8th International Conference on Learning Representations. – 2020. – P. 1–16.
[8] Keskar N.S., McCann B., Varshney L., Xiong C., Socher R. CTRL – A Conditional Transformer Language Model for Controllable Generation // Computing Research Repository. – 2019. – arXiv:1909.05858. – Access mode: https://arxiv.org/abs/1909.05858.
[9] Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts. – 2015. – P. 320–332.
[10] Lester B., Al-Rfou R., Constant N. The Power of Scale for Parameter-Efficient Prompt Tuning // Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. – 2021. – P. 3045–3059.
[11] Li B., Lee-Urban S., Johnston G., Riedl M. O. Story generation with crowdsourced plot graphs // Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence. – 2013. – P. 598–604.
[12] Li X. L., Liang P. Prefix-Tuning: Optimizing Continuous Prompts for Generation // Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing. – 2021. – P. 4582–4597.
[13] Lin B.Y., Zhou W., Shen M., Zhou P., Bhagavatula C., Choi Y., Ren X. CommonGen: A constrained text generation challenge for generative commonsense reasoning // Findings of the Association for Computational Linguistics: EMNLP 2020. – 2020. – P. 1823–1840.
[14] Meister C., Vieira T., Cotterell R. If beam search is the answer, what was the question? // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. – 2020. – P. 2173–2185.
[15] Pascual D., Egressy B., Meister C., Cotterell R., Wattenhofer R. A Plug-and-Play Method for Controlled Text Generation // Findings of the Association for Computational Linguistics: EMNLP 2021. – 2021. – P. 3973–3997.
[16] Prabhumoye S., Black A.W., Salakhutdinov R. Exploring Controllable Text Generation Techniques // Proceedings of the 28th International Conference on Computational Linguistics. – 2020. – P. 1–14.
[17] Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I. Language models are unsupervised multitask learners // OpenAI blog. – 2019. – Vol. 1(8). – Access mode: https://openai.com/blog/better-language-models.
[18] Rose S., Engel D., Cramer N., Cowley W. Automatic keyword extraction from individual documents // Text Mining: Applications and Theory. – 2010. – P. 3–20.
[19] Rosenthal J.A. Qualitative descriptors of strength of association and effect size. Journal of social service Research. – 1996. – Vol. 21(4). – P. 37–59.
[20] Tambwekar P., Dhuliawala M., Martin L.J., Mehta A., Harrison B., Riedl M.O. Controllable Neural Story Plot Generation via Reward Shaping // Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. International Joint Conferences on Artificial Intelligence Organization. – 2019. – P. 5982–5988.
[21] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L., Polosukhin I. Attention is All you Need // Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS). – 2017. – Vol. 30. – P. 6000–6010.
[22] Vychegzhanin S., Kotelnikov E. Collocation2Text: Controllable Text Generation from Guide Phrases in Russian // Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialogue-2022" – Issue 21. – P. 564–576.

[23] Welleck S., Kulikov I., Roller S., Dinan E., Cho K., Weston J. Neural text generation with unlikelihood training // Proceedings of the 8th International Conference on Learning Representations. – 2020. – P. 1–18.
[24] Yang K., Tian Y., Peng N., Klein D. Re3: Generating longer stories with recursive reprompting and revision // Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022). – 2022. – P. 4393–4479.
[25] Yao L., Peng N., Weischedel R., Knight K., Zhao D., Yan R. Plan-and-Write: Towards Better Automatic Storytelling // Proceedings of the AAAI Conference on Artificial Intelligence. – 2019. – Vol. 33(01). – P. 7378–7385.
[26] Zhang H., Song H., Li S., Zhou M., Song D. A Survey of Controllable Text Generation using Transformer-based Pre-trained Language Models // Computing Research Repository. – 2022. – arXiv:2201.05337. Access mode: https://arxiv.org/abs/2201.05337.
[27] Zhang Y., Wang G., Li C., Gan Z., Brockett C., Dolan B. POINTER: Constrained Progressive Text Generation via Insertion-based Generative Pre-training // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). – 2020. – P. 8649–8670.
[28] Zhu Y., Lu S., Zheng L., Guo J., Zhang W., Wang J., Yu J. Texygen: A benchmarking platform for text generation models // The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. – 2018. – P. 1097–1100.