Automatic Text Generation in Macedonian Using Recurrent Neural Networks

Ivona Milanova, Ksenija Sarvanoska, Viktor Srbinoski, and Hristijan Gjoreski(✉)

Faculty of Electrical Engineering and Information Technologies, University of Ss. Cyril and Methodius in Skopje, Skopje, North Macedonia
ivonamilanova22l@gmail.com,
ksenija.sarvanoska@gmail.com,
viktor_srbinoski@hotmail.com,
hristijang@feit.ukim.edu.mk

Abstract. Neural text generation is the process of a training neural network to generate a human understandable text (poem, story, article). Recurrent Neural Networks and Long-Short Term Memory are powerful sequence models that are suitable for this kind of task. In this paper, we have developed two types of language models, one generating news articles and the other generating poems in Macedonian language. We developed and tested several different model architectures, among which we also tried transfer-learning model, since text generation requires a lot of processing time. As evaluation metric we used ROUGE-N metric (Recall-Oriented Understudy for Gisting Evaluation), where the generated text was tested against a reference text written by an expert. The results showed that even though the generate text had flaws, it was human understandable, and it was consistent throughout the sentences. To the best of our knowledge this is a first attempt in automatic text generation (poems and articles) in Macedonian language using Deep Learning.

Keywords: Text generation · Storytelling · Poems · RNN · Macedonian language · NLP · Transfer learning · ROUGE-N

1 Introduction

As the presence of Artificial Intelligence (AI) and Deep Learning has become more prominent in the past couple of years and the fields have acquired significant popularity, more and more tasks from the domain of Natural Language Processing are being implemented. One such task is automatic text generation, which can be designed with the help of deep neural networks, especially Recurrent Neural Networks [16]. Text generation is the process of preparing text for developing a word-based language model and designing and fitting a neural language model in such a way that it can predict the likelihood of occurrence of a word based on the previous sequence of words used in the source text. After that the learned language model is used to generate new text with similar statistical properties as the source text.

S. Gievska and G. Madjarov (Eds.): ICT Innovations 2019, CCIS 1110, pp. 1–12, 2019.
https://doi.org/10.1007/978-3-030-33110-8_1

In our paper, we do an experimental evaluation of two types of word-based language models in order to create a text generation system that will generate text in Macedonian. The first model is generating paragraph of a news article and the other is generating poems. For generating news articles, we also tried implementing transfer learning, but as there are no pre-trained models on a dataset in Macedonian, we used a model that was trained on a dataset in English, so the results were not satisfying. The second model we created was used to generate poetry and was trained on a dataset consisting of Macedonian folk poems. In order to measure how closely the generated text resembles a human written text, we used a metric called ROUGE-N (Recall-Oriented Understudy for Gisting Evaluation), which is a set of metrics for evaluating automatic generation of texts as well as machine translation. With this metric, we got an F1 score of 65%.

2 Related Work

Recent years have brought a huge interest in language modeling tasks, a lot of them being automatic text generation from a corpus of text, as well as visual captioning and video summarization. The burst of deep learning and the massive development of hardware infrastructure has made this task much more possible.

Some of the tasks in this field include automatic text generation based on intuitive model and using heuristics to look for the elements of the text that were proposed by human feedback [1, 2]. Another approach has leaned towards character-based text generation using a Hessian-free optimization in order to overcome the difficulties associated with training RNNs [3]. Text generation using independent short description has also been one topic of research. The purpose of this paper has been to describe a scene or event using independent descriptions. They have used both Statistical Machine Translation and Deep Learning to present text generation in two different manners [4]. The other kind of text generation application has been designing text based interactive narratives. Some of them have been using an evolutionary algorithm with an end-to-end system that understands the components of the text generation pipeline stochastically [5] and others have been using mining of crowd sourced information from the web [6, 7].

Many papers have also focused on visual text generation, image captioning and video description. One recent approach to image captioning used CNN-LSTM structures [8, 9]. Sequence-to-sequence models have been used to caption video or movie contents. They are using an approach where the first sequence encodes the video and the second decodes the description [10, 11].

The idea behind document summarization has been used on video summarization where instead of extracting key sentences, key frames or shots are selected [12].

Visual storytelling is the process of telling a coherent story about an image set. Some of the works covering this include storyline graph modeling [13] and unsupervised mining [14].

Another state-of-the-art approach includes using hierarchically structured reinforcement learning for generating coherent multi-sentence stories for the visual storytelling task [25].

However, all of these approaches include text generation in English where the amount of available data is enormous. Our paper focuses on creating stories in Macedonian using data from Macedonian news portals and folk poetry as well as exploration of different model architectures in order to get the best result, regarding comprehensibility and execution time.

3 Dataset and Preprocessing

The first dataset that we used was gathered from online news portals and consisted of around 2.5 million words. The data was collected with the help of a scraper program we wrote in .NET Core using the C# programming language. The program loads the web page from a given Uniform Resource Locator (URL) and then looks at the html tags. When it finds a match for the html tags that we have given the program, it takes their content and writes it to a file.

The second dataset consisted of a collection of Macedonian poetry written by various Macedonian writers [17] and was made up of roughly 7 thousand words. The data was not clean so we had to do a fair amount of data preprocessing.

The collected datasets are publically available at https://github.com/Ivona221/MacedonianStoryTelling.

In order to prepare the data that we collected to be suitable to enter the algorithm and to simplify the task of the algorithm when it starts learning, we had to do a considerable amount of data cleaning. For this purpose, we created a pipeline in C#, which in the end gave us a clean dataset to work on. The first step in the algorithm was to remove any special characters from the text corpus including html tags that were extracted from the websites along with the text, JavaScript functions and so on. We also had to translate some of the symbols into text like dollar signs, degree signs and mathematical operators in order to have the least amount of unique characters for the algorithm to work with. The next step was to translate all the English words if there existed a translation or remove the sentences where that was not the case. Next, we had to separate all the punctuation signs from the words with an empty space in order for the algorithm to consider them as independent words. The last and one of the most important steps in this pipeline was creating a custom word tokenizer. All the existing word tokenizers were making a split on an empty space. However, they do not take into consideration the most common word collocations as well as words containing dash, name initials and abbreviations. Our algorithm was taking these words as one. The abbreviations were handled by using a look-up table of all the Macedonian abbreviations, as well as the most common collocations and the initials were handled by searching for specific patterns in text like capital letter-point-capital letter (Fig. 1).

Fig. 1. Data preprocessing flow

4 Language Model Architecture

We have trained two types of models that work on the principle of predicting the next word in a sequence, one for news generation and one for poem generation. The language models used were statistical and predicted the probability of each word, given an input sequence of text. For the news generation model, we created several different variations, including a transfer learning approach (Fig. 2).

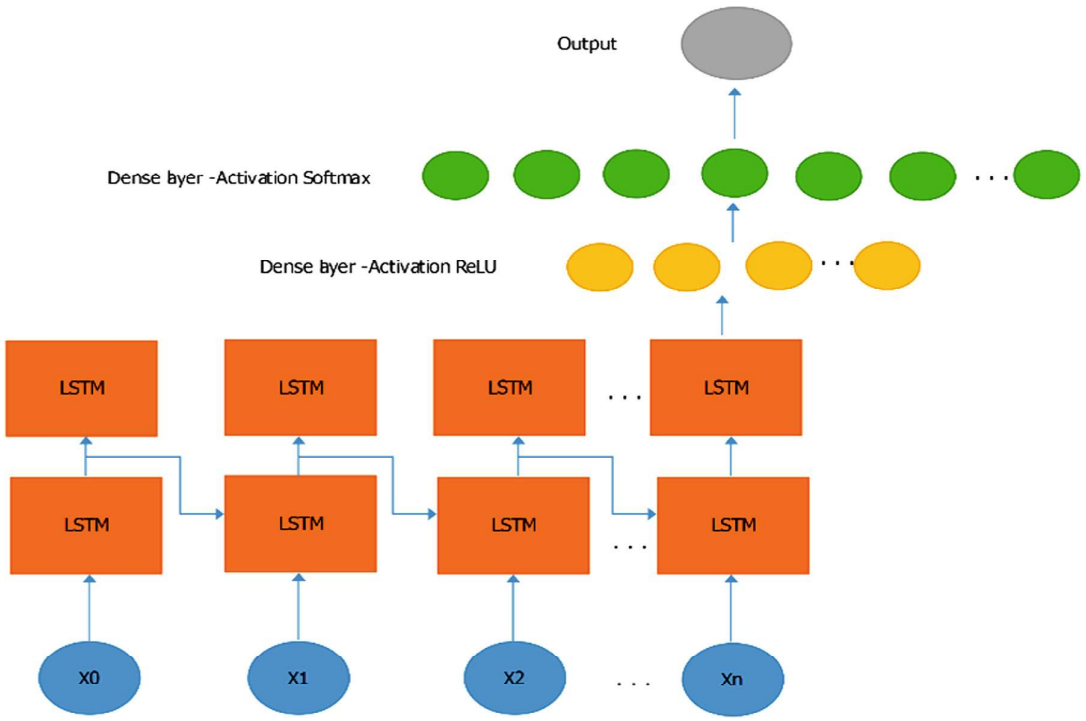

Fig. 2. Language model architecture

4.1 News Article Generation

The first approach for news generation was trained on news articles and used a sequence of hundred words as input. It then generates one word based on that sequence, meaning the word with the biggest probability of appearing next. In the next time step it adds the generated word to the sequence of a hundred words and cuts out the very first word, meaning that it once again makes a sequence with a length of one hundred. It then feeds this new sequence to itself as input for the next time step and continues doing so until it generates the preset amount of words.

We tried out multiple architectures and the best results were acquired from the following architecture. The neural network was an LSTM (Long-Short Term Memory) recurrent neural network with two LSTM layers and two dense layers and also, we tried a variation with a dropout layer in order to see how that affects the performance. The first LSTM layer consists of 100 hidden units and 100 timesteps and is configured to give one hidden state output for each input time step for the single LSTM cell in the

layer. The second layer we added was a Dropout layer with a dropout rate of 0.2. They key idea behind adding a dropout layer is to prevent overfitting. This technique works by randomly dropping units (along with their connections) from the neural network during training. This prevents units from co-adapting too much [18]. The next layer is also an LSTM layer with 100 hidden units. Then we added a Dense layer which is a fully connected layer. A dense layer represents a matrix vector multiplication. The values in the matrix are the trainable parameters, which get updated during back-propagation. Assuming we have an n-dimensional input vector u (in our case 100-dimensional input vector) presented with the formula:

$$u \in R^{n*1} \tag{1}$$

We get an m-dimensional vector as output.

$$u^{T.W} = W \in R^{n*m} \tag{2}$$

A dense layer thus is used to change the dimensions of the vector. Mathematically speaking, it applies a rotation, scaling, translation transform to your vector. The activation function meaning the element-wise function we applied on this layer was ReLU (Rectified Linear Units). ReLU is an activation function introduced by [19]. In 2011, it was demonstrated to improve training of deep neural networks. It works by thresholding values at zero, i.e. f(x) = max (0, x). Simply put, it outputs zero when x < 0, and conversely, it outputs a linear function when x $\geq$ 0. The last layer was also a Dense layer, however with a different activation function, softmax. Softmax function calculates the probability distribution of the event over 'n' different events. In a general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs. As the loss function or the error function we decided to use sparse categorical cross entropy. A loss function compares the predicted label and true label and calculates the loss. With categorical cross entropy, the formula to compute the loss is as follows:

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o.c}) \tag{3}$$

where,

- M – number of classes
- log – the natural log
- y – binary indicator (0 or 1) if class label c is the correct classification for observation o
- p – predicted probability observation o is of class c

The only difference between sparse categorical cross entropy and categorical cross entropy is the format of true labels. When we have a single-label, multi-class classification problem, the labels are mutually exclusive for each data, meaning each data entry can only belong to one class. Then we can represent y_true using one-hot embeddings. This saves memory when the label is sparse (the number of classes is very large).

As an optimizer we decided to use Adam optimizer (Adaptive Moment Estimation) which is method that computes adaptive learning rates for each parameter. It is an algorithm for first-order gradient-based optimization of stochastic objective functions. In addition to storing an exponentially decaying average of past squared gradients vt like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients m_t, similar to momentum [20]. As an evaluation metric we used accuracy.

For the second approach, we tried using a transfer learning method using the pre trained model word2vec, which after each epoch generated as many sentences as we gave it starting words on the beginning. Word2vec is a two-layer neural net that processes text. On the pre-trained model, we added one LSTM layer and one Dense layer. When using a pretrained model the embedding or the first layer in our model is seeded with the word2vec word embedding weights. We trained this model on 100 epochs using a batch size of 128. However, as the pre-trained models are trained using English text, this model did not give us satisfying results and that is the reason why this models results will not be observed in this paper [22].

4.2 Poem Generation

This model has only slight differences from the first LSTM network we described. It is an LSTM neural network as well, with two LSTM layers and one dense layer. Having experimented with several different combinations of parameters we decided on the following architecture. The first LSTM layer is made up of 150 units, then we have a Dropout layer with a dropout rate of 0.2 so that we can reduce overfitting. The second LSTM layer is made up of 100 units and it has another Dropout layer after it with a dropout rate of 0.2. At last, we have a dense layer with a softmax activation function, which picks out the most fitting class or rather word for the given input.

In this model we have decided on the loss function to be categorical cross entropy and as an optimizer once again we use the Adam optimizer.

Another thing that we do differently with the second model is the usage of a callback function EarlyStop [21]. A callback is a set of functions that are applied at certain stages of the training procedure with the purpose of viewing the internal states and statistics of the model during training. EarlyStop helps lessen the problem of how long to train a network, since too little training could lead to under fitting to the train and test sets while too much training leads to overfitting. We train the network on a training set until the performance on a validation set starts to degrade. When the model starts learning the statistical noise in the training set and stops generalizing, the generalizing error will increase and signal overfitting. With this approach during the training after every epoch, we evaluate the model on a holdout validation dataset and if this performance starts decaying then the training process is stopped. Because we are certain that the network will stop at an appropriate point in time, we use a large number of training epochs, more than normally required, so that the network is given an opportunity to fit and then begin to over fit to the training set. In our case, we use 100 epochs.

Early stopping is probably the oldest and most widely used form of neural network regularization.

5 Text Generation

As mentioned before the first language model is fed a sequence of hundred words and to make this possible, a few steps in the text preprocessing need to be taken. With the tokenizer we first vectorize the data by turning the text into a sequence of integers, each integer being the index of a token in a dictionary. We then construct a new file which contains our input text but makes sure to have one hundred words per each line. In the text generation process we randomly select one line from the previously created file for the purpose of generating a new word. We then encode this line of text to integers using the same tokenizer that used when training the model. The model then makes a prediction of the next word and gives an index of the word with the highest probability which we must look up in the Tokenizers mapping to retrieve the associated word. We then append this new word to the seed text and repeat the process.

Considering that the sequence will eventually become too long we can truncate it to the appropriate length after the input sequence has been encoded to integers.

6 Results

As we mentioned before we trained two different main models one for news article generation and another one for poems.

The accuracy and loss for this kind of task are calculated on the train set since one cannot measure the correctness of a story, therefore test set cannot be constructed. In order to evaluate the result we compared it against a human produced equivalent of the generated story.

Regarding the news generation model, we tried two variations, one with dropout and one without dropout layers and tested how that affected the training accuracy and loss. Both variations were trained on 50 epochs, using a batch size of 64. Comparing the results, adding dropout layers improved accuracy and required shorter training time (Figs. 3 and 4).

The poem generation model was trained on 100 epochs using a batch size of 64. This model also included dropout layers (Figs. 5 and 6).

In order to evaluate how closely the generated text resembles a human written text, we used ROUGE-N metric. It works by comparing an automatically produced text or translation against a set of reference texts, which are human-produced. The recall (in the context of ROUGE) refers to how much of the reference summary the system summary is recovering or capturing. It can be computed as:

$$\frac{number\_of\_overlapping\_words}{total\_words\_in\_reference\_text} \tag{4}$$

The precision measures how much of the system summary was in fact relevant or needed. It is calculated as:

$$\frac{number\_of\_overlapping\_words}{total\_words\_in\_system\_generated\_text} \tag{5}$$

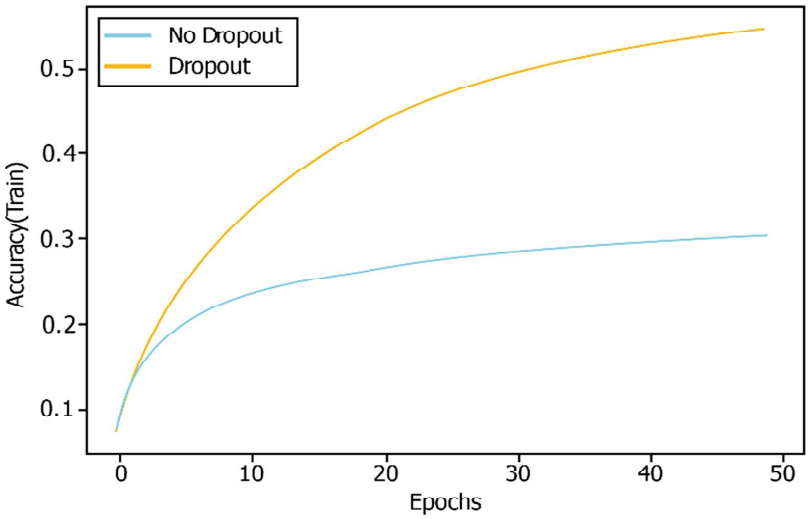

Fig. 3. Train accuracy, comparison with and without dropout layer

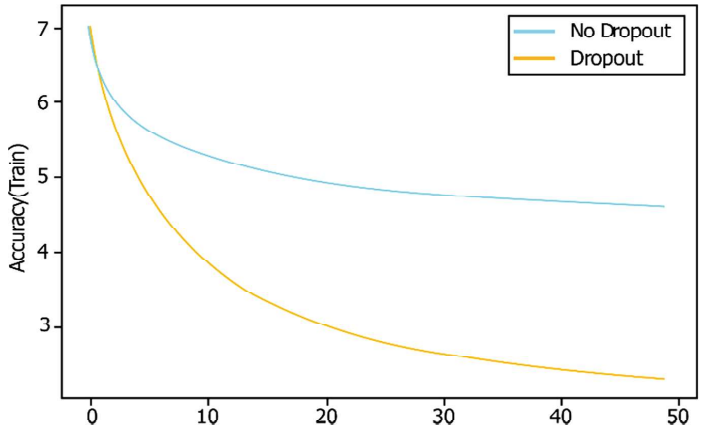

Fig. 4. Train loss, comparison with and without dropout layer

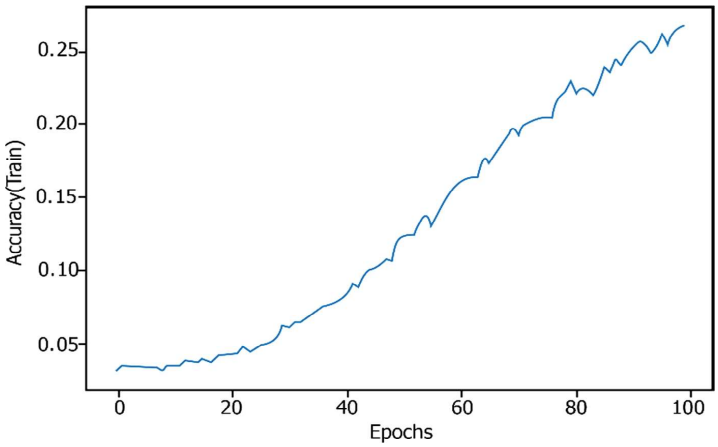

Fig. 5. Train accuracy

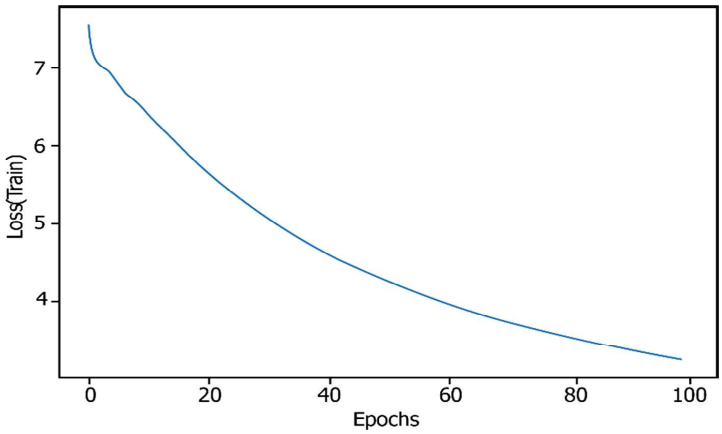

Fig. 6. Train loss.

Using the precision and recall, we can compute an F1 score with the following formula:

$$F1 = \frac{precision * recall}{precision + recall} * 2 \tag{6}$$

In our case we used ROUGE-1, ROUGE-2 and ROUGE-L.

- ROUGE-1 refers to overlap of unigrams between the system summary and reference summary
- ROUGE-2 refers to the overlap of bigrams between the system and reference summaries
- ROUGE-L measures longest matching sequence of words using LCS (Longest Common Subsequence). An advantage of using LCS is that it does not require consecutive matches, but in-sequence matches that reflect sentence level word order. Since it automatically includes longest in-sequence common n-grams, you do not need a predefined n-gram length.

The reason one would use ROUGE-2 over or in conjunction with ROUGE-1, is to also show the fluency of the texts or translations. The intuition is that if you follow the word orderings of the reference summary more closely, then your summary is actually more fluent [24].

The results from the ROUGE-N metric are shown in Tables 1 and 2:

Table 1. Results for the news generation model

	Precision %	Recall %	F1-score %
ROUGE-1	66.67	76.92	71.43
ROUGE-2	35.85	57.58	44.19
ROUGE-L	66.67	76.92	70.71

Table 2. Results for the poem generation model

	Precision %	Recall %	F1-score %
ROUGE-1	47.89	46.58	47.22
ROUGE-2	21.05	21.28	21.16
ROUGE-L	40.85	39.73	40.26

In the end we present a sample generated from each of the models:

- News generation model:

...ќе се случи и да се користат на корисниците, а и да се случи, се наоѓа во играта, а и да се случи, се уште не се случува во Македонија. Во моментов, во текот на државата се во прашање, се во прашање, но не и да се случи, но и за да се случи, се наведува во соопштението на САД. Во текот на државата се во прашање, се во прашање, и да се случи, се уште не и корупциски скандали, се наоѓа во водата и...

- Poem generation model:

Во залула време тамо - од прангите звекот, оро на земниот
земниот роден кат, - трендафил во одаја, плиснала врева. магла
на очиве моја! жетва најбогата цвета, оро на кај, шар.
јазик љубовта наша делата в тих. рака јазик падна, -
стиснат сите љуѓе - ноќ, убава, красна, дише рој, писнат,
најде на туѓина - чемер во одаја, шар. јазик идеш
праг, роден кат, и скитник со здржана туѓи оро син
стиснала - ПОЕМА - нок, убава, красна, дише на робја,
шар. издржа сите љуѓе - ноќ, убава, красна, дише рој,
писнат, мајка, кат, - развива ора, во онаа вечер, в
очи очиве носам, у секоа доба. оро ќе сретнам, најде.

7 Conclusion

In this paper we present a solution to the problem of automatic text generation in Macedonian language. To the best of our knowledge this is a first attempt in automatic text generation (poems and articles) in Macedonian language using Deep Learning.

We made attempts with two types of models, the first for news generation and the second for poem generation. We also tried a transfer learning model using word-2-vec, however the results were not satisfying. Excluding the network where we used transfer learning, we got promising results. The first model was able to generate a text of a hundred words, including punctuation marks. It used syntax rules correctly and put punctuation marks where needed. Even though the generated output of the poem model was nonsensical, it was still a clear indication that the network was able to learn the style of the writers and compose a similar looking piece of work.

There is, of course, a lot left to be desired and a lot more can be done to improve upon our work, such as using a more cohesive dataset. Because our dataset was created by putting different news articles together, there is no logical connection from one news article to the other, which resulted in our model not having an output that made much sense. The same point can apply to our dataset of poems where each poem was standalone and there was no connection from one to the other. Another suggestion for achieving better results is adding more layers and units to the networks, keeping in mind that as the size of the neural network gets bigger, so do the hardware requirements and training time needed [23].

Acknowledgment. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

1. Bailey, P.: Searching for storiness: story-generation from a reader's perspective. In: Working Notes of the Narrative Intelligence Symposium (1999)
2. PÉrez, R.P.Ý., Sharples, M.: MEXICA: a computer model of a cognitive account of creative writing. J. Exp. Teor. Artif. Intell. **13**, 119–139 (2001)
3. Sutskever, I., Martens, J., Hinton, G.E.: Generating text with recurrent neural networks. In: Proceedings of the 28th International Conference on Machine Learning (ICML-2011), pp. 1017–1024 (2011)
4. Jain, P., Agrawal, P., Mishra, A., Sukhwani, M., Laha, A., Sankaranarayanan, K.: Story generation from sequence of independent short descriptions. In: Proceedings of Workshop on Machine Learning for Creativity, Halifax, Canada, August 2017 (SIGKDD 2017) (2017)
5. McIntyre, N., Lapata, M.: Learning to tell tales: a data-driven approach to story generation. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, Association for Computational Linguistics, vol. 1, pp. 217–225 (2009)
6. Li, B., Lee-Urban, S., Johnston, G., Riedl, M.: Story generation with crowdsourced plot graphs. In: AAAI (2013)
7. Swanson, R., Gordon, A.: Say anything: a massively collaborative open domain story writing companion. Interact. Storytelling **2008**, 32–40 (2008)
8. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: a neural image caption generator. In: CVPR (2015)
9. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. In: ICML (2015)
10. Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., Saenko, K.: Sequence to sequence-video to text. In: ICCV (2015)
11. Pan, Y., Mei, T., Yao, T., Li, H., Rui, Y.: Jointly modeling embedding and translation to bridge video and language. In: CVPR (2016)
12. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: EMNLP (2015)
13. Kim, G., Xing, E.P.: Reconstructing storyline graphs for image recommendation from web community photos. In: CVPR (2014)
14. Sigurdsson, G.A., Chen, X., Gupta, A.: Learning visual storylines with skipping recurrent neural networks. In: ECCV (2016)

15. Glorianna Jagfeld, S.J.: Sequence-to-sequence models for data-to-text natural language (2018)
16. Sutskever, I.: Generating text with recurrent neural networks. In: 28th International Conference on Machine Learning (ICML-2011), pp. 1017–1024 (2011)
17. Racin, K.: Beli mugri, pp. 3–33. Makedonska kniga, Skopje (1989)
18. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
19. Hahnloser, R.H., Sarpeshkar, R., Mahowald, M.A., Douglas, R.J., Seung, H.S.: Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature **405** (6789), 947 (2000)
20. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs. LG], December 2014
21. Montavon, G., Orr, Geneviève B., Müller, K.-R. (eds.): Neural Networks: Tricks of the Trade. LNCS, vol. 7700. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8
22. Mikolov, T., et al.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
23. Pradhan, S.: Exploring the depths of recurrent neural networks with stochastic residual learning (2016)
24. Lin, C.-Y.: ROUGE: a package for automatic evaluation of summaries. In: ACL Workshop: Text Summarization Braches Out 2004, p. 10 (2004)
25. Huang, Q., Gan, Z., Celikyilmaz, A., Wu, D., Wang, J., He, X.: Hierarchically structured reinforcement learning for topically coherent visual story generation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 8465–8472, July 2019