Model for generating jokes

Aleksey Grishin, Maksim Korotkov

May 2023

Abstract

This project will be about the task of generating jokes. Firstly, we will look at related works. Secondly, describe the dataset which we used. Finally, describe the model and show the results. You can also look at the code of the project: `https://github.com/shlexsoid/NLP-Joke-Generator`.

1 Introduction

Humor has always played an important part in human communication. Laughing at one another's jokes and remarks often shows understanding and empathy. Humorous communication in human-machine interactions can strengthen the relationship between humans and virtual assistants. Large companies such as Google and Apple are already hiring comedy writers to make their virtual assistants funnier for this purpose [1][2]. The responses of their virtual assistants are thus written beforehand, rather than generated by the program itself. Generating humor is a complex task in the domain of machine learning, and it requires the models to understand the deep semantic meaning of a joke in order to generate new ones. Such problems, however, are difficult to solve due to a number of reasons, one of which is the lack of a database that gives an elaborate list of jokes.

1.1 Team

Maksim Korotkov prepared this document and took part in model preparing. **Aleksey Grishin** prepared a parser for the dataset and took part in model preparing.

2 Related Work

Computational humor is a field of artificial intelligence linking humor with computational artifacts. Many advancements have been made over the years in this field in terms of linguistic theories as well as in computer programs capable of

generating or detecting specific types of humor. These programs are, for example, capable of creating punning riddles [3][4][5], generating analogies [6], detecting innuendos and one-liners [7][8] and more [9][10][11][12][13][14][15]. However, there has been little research on making the generated humor adapt to the user or adapt over time. Most humor generation research focuses on making jokes based on fixed rule sets, which allow the efficient creation of millions of jokes or the discovery of jokes according to the researcher's assumptions. Such programs do not learn how humorous certain users might find certain jokes nor do they improve themselves over time.

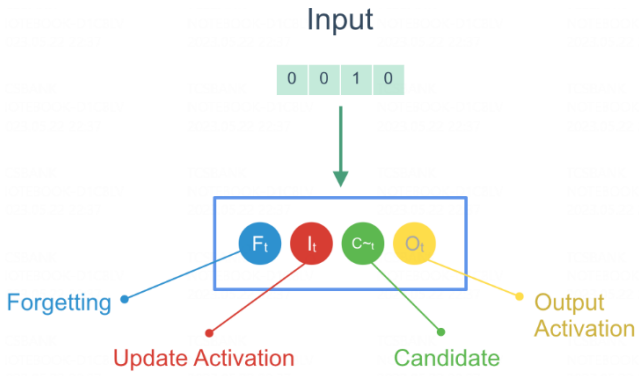
Moreover, there is a kaggle competition in this sphere `https://www.kaggle.com/datasets/abhinavmoudgil95/short-jokes` which shows the interest to the topic of generating jokes in modern society.

By the way, we should stress that all previous works are about English jokes. In our project we are going to make a model of generating jokes in Russian. We will also try to make model for the dataset from the kaggle competition above.

3 Model Description

In this project we are going to use a LSTM model for Russian dataset. Long Short Term Memory Models (LSTMs) were invented to try and solve this problem. Let's take a look into how they work.

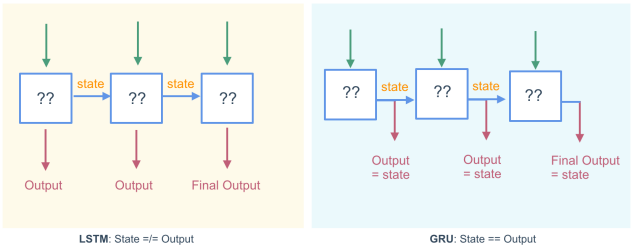
Just like simple RNN's, LSMTs take in a vector and output a vector. Unlike RNNs however, the state passed to the next iteration is different to the output at each step. LSTMs do a bunch of precalaculations before they decide what to output and what to pass along as state. They perform four independent calculations using perceptrons at each step, and use the result of those calculations to decide the new state and the output. The four calculations are Forgetting (Ft), Update Activation (It), Candidate (C t) and Output Activation (Ot).



As for the English dataset, we will use GRU model. What is it? Gated Recurrent Units (GRU) were invented in 2014, and are a simplified version of LSTMs that can train a little faster with often no loss in performance. There

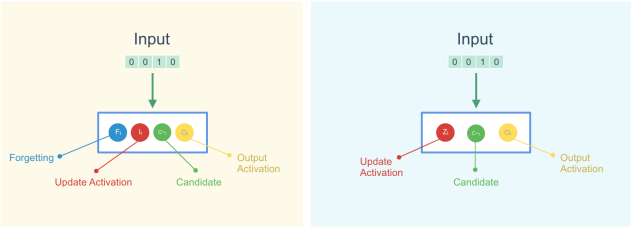
are 2 big differences between a GRU and an LSTM. How does it work?

Firstly, the output values are the same values as the state that it passes into the next step.



LSTM: State =/= Output **GRU**: State == Output

Secondly, there are only three computations that we are doing, not four. If we look at LSTMs, we have a different vector for what to forget and what to update. This may seem a little redundant - if we forget the previous value, we want to update it, and if we don't forget the previous value we don't want to update it. GRUs merge these two vectors into a single vector - the forget value is just (1 - the update activation).



LSTM: 4 vectors **GRU**: 3 vectors

After that, it is necessary to explain the algorithm of the function which generate jokes using our GRU model:

1. the initial state of the hidden layer is random, which makes it possible to randomly generate words provided that the first word is the mark of the beginning of the line

2. the state of the hidden layer is updated based on the given first words of the text

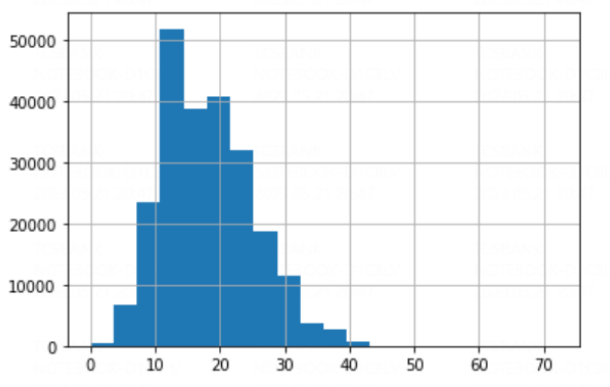
3. based on the last word and state of the hidden layer, a new word and state of the hidden layer is generated

4 Dataset

As we said before, we are going to try models on Russian and English jokes. So we need two datasets for it.

Firstly, let's look on English dataset, containing 231,657 jokes. Length of jokes ranges from 10 to 200 characters. Each line in the file contains a unique ID and joke. During the preprocessing we:

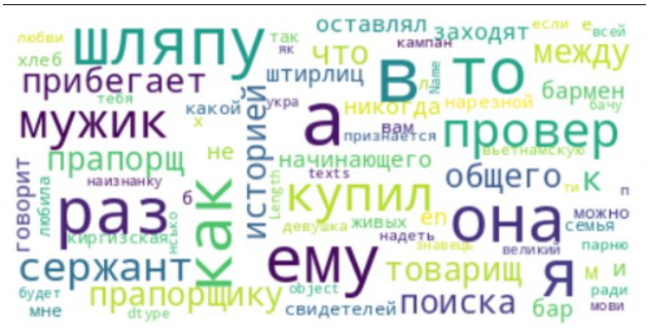
- delete the jokes that a less than 10 words and higher than 25 words. This step was done because of the distribution of the lenghts of jokes, which is presented on the histogram below.



- change the text to lowercase
- remove digits
- remove punctuation

After the preprocessing the length of vocab was 2548 tokens. Among the most common words were: what, you, i etc.

Secondly, let's look on Russian dataset. This dataset was collected on our own by parsing jokes from `https://vk.com/jumoreski`. The dataset was parsed by VK API, the code of parsing is also presented on the github. The dataset consists of 15 085 jokes. The process of preprocessing was similar to the preprocessing on the English dataset. The length of vocab is 959 words. The most common words are presented on the word cloud below.



5 Experiments

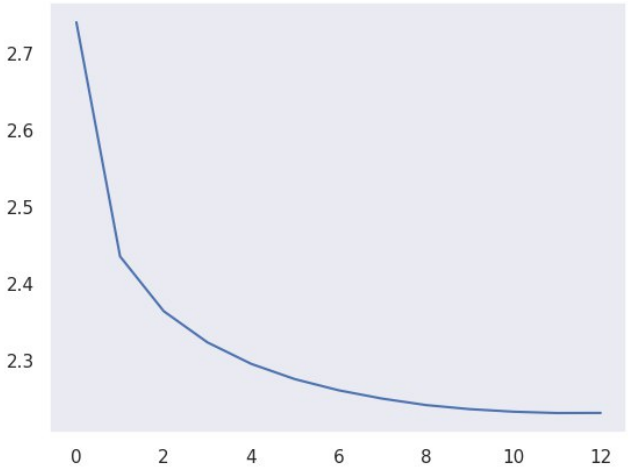
5.1 Metrics

To evaluate the approach we looked at the loss during the epochs.

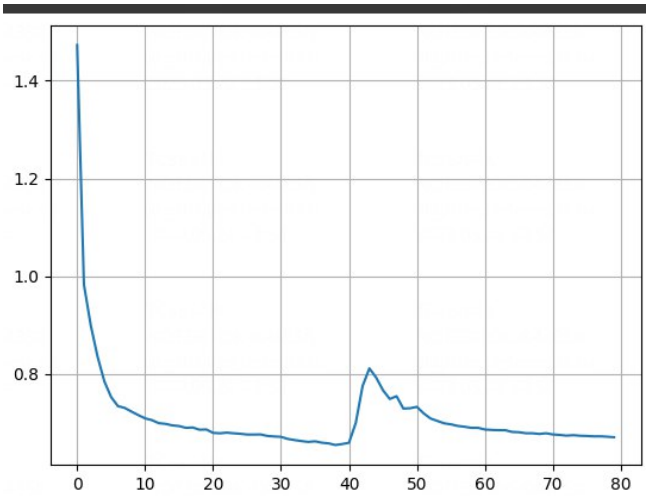
5.2 Experiment Setup

We were evaluating the model on 10 epochs. The plot of loss on epochs is presented below:

- English dataset



- Russian dataset



6 Results

In this section we are going to present some jokes generated by our models
- GRU model (English dataset):
1) congratulations if you re fat
2) you never wanna grow old
- LSTM model (Russian dataset):
1) Заходит жена
2) пупа и лупа
The results are bad, we need more dataset to achieve better results

7 Conclusion

To sum up, we should highlight the main achievments of the project:
- we found out works about generating jokes
- we made our own Russian dataset of jokes by parsing VK.
- tried GRU model for generating English and Russian jokes
- laughed at jokes of our model.

8 References

1. Heate, B.: Google is looking to creative writers and comedians to help humanize assistant. https://techcrunch.com/2016/10/10/google-laughsistant/ (2016)

2. Zwaag, G.V.D.: Apple zoekt een grappenmaker voor siri: heb jij genoeg humor om te helpen? https://www.iculture.nl/nieuws/ siri-vacature-grappen-bedenken/ (2016)

3. Binsted, K., Ritchie, G.: An implemented model of punning riddles. CoRR abs/cmp-lg/9406022 (1994)

4. Manurung, R., Ritchie, G., Pain, H., Waller, A., Mara, D., Black, R.: The construction of a pun generator for language skills development. Applied Artificial Intelligence 22(9) (2008) 841–869

5. Agustini, T., Manurung, R.: Automatic evaluation of punning riddle template extraction. In: ICCC. (2012) 134–139

6. Petrovi´c, S., Matthews, D.: Unsupervised joke generation from big data. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Sofia, Bulgaria, Association for Computational Linguistics (August 2013) 228–232

7. Kiddon, C., Brun, Y.: That's what she said: Double entendre identification. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL), Portland, OR, USA (June 2011) 89–94 http://dl.acm.org/ citation.cfm?id=2002756ACM ID: 2002756.

8. Mihalcea, R., Strapparava, C.: Making computers laugh: Investigations in automatic humor recognition. In: HLT/EMNLP 2005, Human Language

Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada. (2005) 531–538

9. Stock, O., Strapparava, C.: Hahacronym: Humorous agents for humorous acronyms. Humor-International Journal of Humor Research 16(3) (2003) 297–314

10. Venour, C.: The computational generation of a class of puns. Master's thesis, Queen's University, Kingston, Ontario (1999)

11. Valitutti, A., Toivonen, H., Doucet, A., Toivanen, J.M.: "let everything turn well in your wife": Generation of adult humor using lexical constraints. In: ACL (2), The Association for Computer Linguistics (2013) 243–248

12. Chandrasekaran, A., Parikh, D., Bansal, M.: Punny captions: Witty wordplay in image descriptions. CoRR abs/1704.08224 (2017)

13. Justin McKay, B.M.: Generation of idiom-based witticism to aid second language learning. Proceedings of April Fools' Day Workshop on Computational Humour (TWLT 20) (April 2002) 77–87

14. Taylor, J.: Computational Recognition of Humor in a Focused Domain. University of Cincinnati (2004)

15. Shahaf, D., Horvitz, E., Mankoff, R.: Inside jokes: Identifying humorous cartoon captions. In: KDD, ACM Press (2015) 1065–1074