Poem Generation - Milestone I

Jakub Bilski, Jakub Brojacz, Jakub Kosterna

March-April 2022

Contents

1 Literature Analysis

1. Oliveira, Hugo. "Automatic generation of poetry: an overview." Universidade de Coimbra (2009)

The oldest article analyzed by us includes generators based on existing poems with the classic approach - no deep learning methods used. The methodologies used there contain among others creating templates and then filling them with matching words and making rules on how to match words to generate matching lines. In addition, the paper included such approaches as creating structures of matching words, word accents, words sorted by subjects, grammatical templates and more. Also the evolutionary algorithm is mentioned.

2. Van de Cruys, Tim. "Automatic poetry generation from prosaic text." Proceedings of the 58th annual meeting of the association for computational linguistics (2020)

The publication presents encoder-decoder architecture, generating poems based on Prosaic Text. The model is made up of gated recurrent units (GRUs) - almost the same as in our Jupyter Notebook. The system is exclusively trained on standard, non-poetic text, and its output is constrained in order to confer a poetic character to the generated verse. The model has been trained on billions of words extracted from generic web texts; its rhyming knowledge has been extracted from Wiktionary, and it automatically learned an elementary notion of sense by looking at the context of words. Even though it only uses standard, non-poetic text as input, the system yields state of the art results for poetry generation.

3. Tanel Kiis, Markus Kängsepp. "Generating Poetry using Neural Networks." (2018)

The paper describes LSTM and Variational Autoencoders methods. LSTM (both word-by-word and sign-by-sign) can figure out grammar, but poems are without meaning; on the other hand, VAE focuses on repeating phrases commonly occured in training data, but it is able to generate at least some poems that could interpreted as a original and coherent poetry. Most of the poems were collected from two sites: Poem Hunter and Poetry Foundation, extracting around 150 000 poems written by top 500 poets.

4. Tikhonov, Aleksey, and Ivan P. Yamshchikov. "Sounds Wilde. Phonetically extended embeddings for author-stylized poetry generation." Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology (2018)

Using a version of the language model with extended phonetic and semantic embeddings to generate poetry, the paper showed that phonetics makes a comparable contribution to overall model performance as author information. Phonetic information was shown to be important for English, and people tend to attribute machine-generated texts to the target author. In the paper, an LSTM with extended phonetic and semantic embeddings and quantify the quality of the obtained stylized poems both subjectively through a survey and objectively with BLEU metrics is presented. Also, it is showed that phonetic information plays key role in a author-stylized poetry generation.

2 Description of data

Data is taken from Poems kaggle dataset. The poems there are categorized by the form (e.g. haiku, sonnet, etc.) or topic (love, nature, joy, peace, etc.).

In order to improve our model's results, poems from the Project Gutenberg may also be included. A free eBook library consists of about texts, which can be useful if aforementioned kaggle dataset will prove to be insufficient.

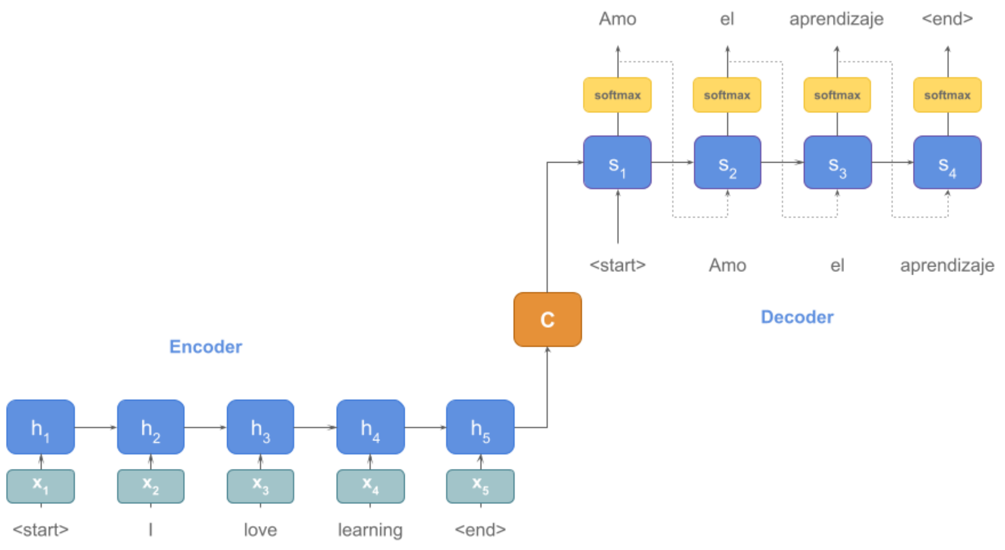

Figure 1: Encoder-decoder schema - in term of translating

Data is divided by type of poem and by subject, so we can check how our model will behave when only selected types of poems are present in training data.

The dataset's site contains also two potentially helpful notebooks by other kaggle users - solutions *Poem Generation using FastAI* and *Starter: Poems Dataset (NLP)* have been briefly reviewed by us so far, but will probably be useful for writing further code.

3 Exploratory Data Analysis

We examined the distribution of line lengths in the selected data set. The results are presented on Fig. 4. The distribution is almost symmetrical around the most common value of 7 words per line, with a quickly diminishing tail in the higher values. However, there are some examples of lines as long as 30 words.

Fig. 3 presents histogram of the word occurrences in the whole data set. After presenting on a log-log scale, the plot is well approximated by a straight line.

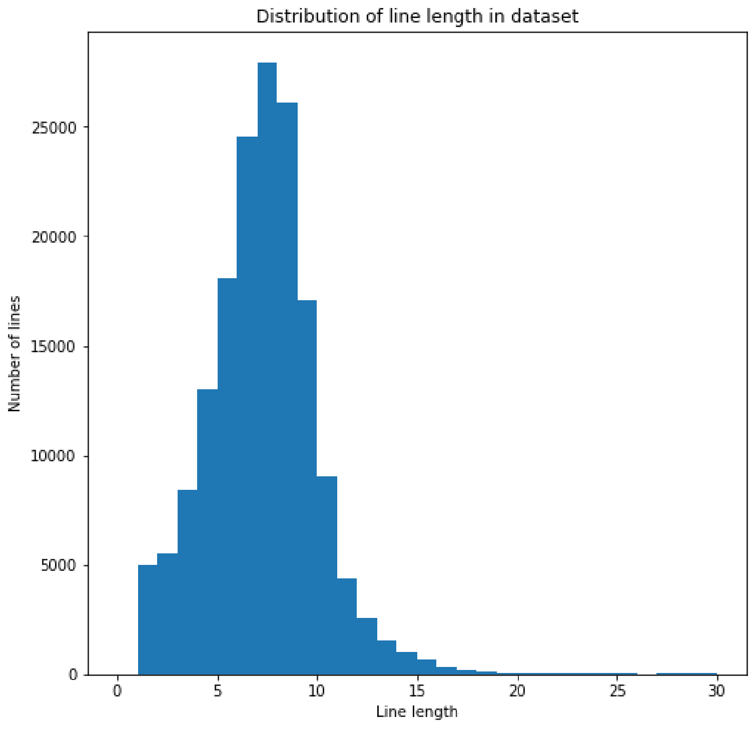

Figure 2: Distribution of line length in dataset, measured in words

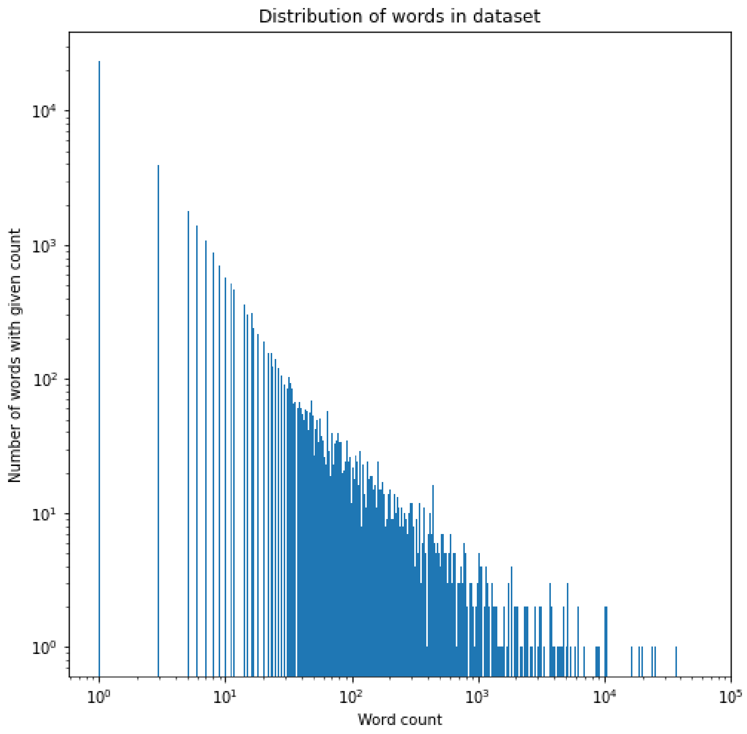

Figure 3: Distribution of words in dataset

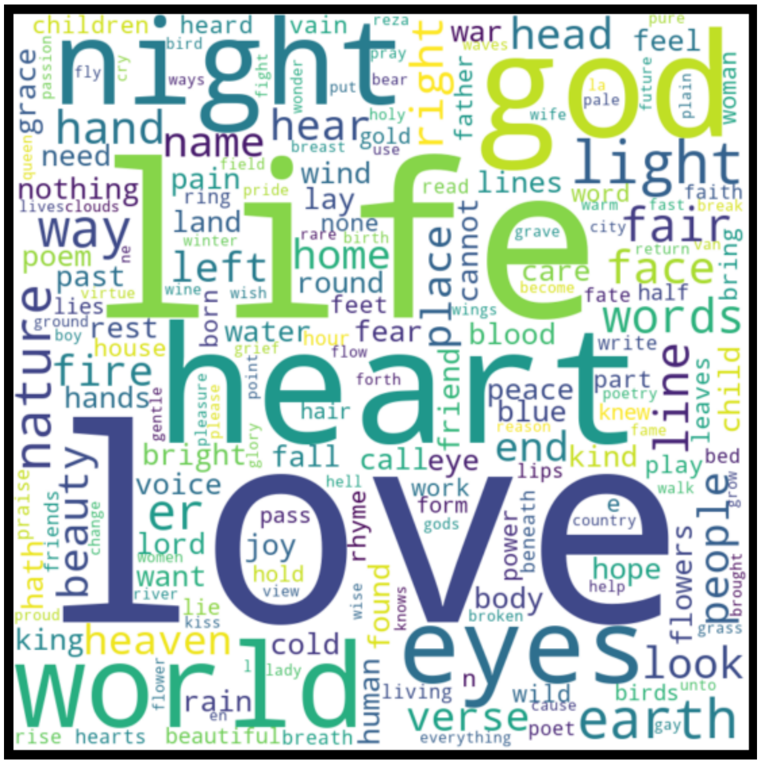

Figure 4: Visualization of the most popular nouns

Fig. 5 contains information about the 20 most commonly occurring words.

The data set is by default divided into subsets containing poems of the same type. Fig. 6 presents information about the mean number of words and lines for the poem types present in the data set in the highest abundance. This table also contains information about the number of detected rhyme patterns. The results suggest that not all rhymes were detected, e.g. each italian-sonnet should contain at least two abba rhymes. The lower value suggests that there will be a need for slant rhymes detection, as the current detection method is probably too strict.

order	word	count
1	love	4057
2	life	2387
3	day	1983
4	heart	1940
5	time	1862
6	man	1813
7	god	1491
8	night	1404
9	world	1360
10	eyes	1294
11	light	1238
12	men	1127
13	soul	1056
14	earth	1055
15	death	1014
16	way	985
17	nature	955
18	mind	954
19	things	923
20	sun	909

Figure 5: Most popular nouns in dataset

4 Solution Concept

Our application will be naturally developed in Python - this decision was made because of the largest number of tools adapted to the problems of natural language processing in this language, and moreover its capabilities meet the needs of writing algorithms based on the cited literature.

We will use **encoder-decoder architecture**, with concept of generating text line-by-line. From set of generated lines we will take the best fit based on rhymes, accents and overall cohesion with the rest of the text. We intent to choose the best objective function that will optimize the creation of poems resembling those created by a living person.

Given the availability and also the thorough testing, we are likely to use a solution involving the pretrained BERT model. While we considered building the model from scratch, it is important to keep in mind that our dataset is too small to teach such a big system. Another problem is a big number of words appearing

poem type	files	words	lines	rhymes		
				aa	abab	abba
epigram	100	45.13	6.63	1.04	0.16	0.04
triolet	100	64.62	9.68	1.3	0.37	0.04
cavatina	100	73.0	12.6	0.88	0.0	0.0
allegory	100	95.5	16.36	0.87	0.16	0.04
tetractys	100	36.72	11.52	0.13	0.0	0.0
italian-sonnet	100	129.59	15.17	2.18	0.17	1.57
free-verse	100	126.86	18.31	0.54	0.07	0.06
tanka	100	33.89	7.68	0.05	0.0	0.0
clerihew	100	26.09	4.55	1.33	0.05	0.03
syllabic-verse	100	96.91	14.8	0.33	0.01	0.0
cinquain	100	22.38	7.23	0.04	0.0	0.0
acrostic	100	95.71	14.28	1.59	0.16	0.33
ballade	100	214.38	31.86	2.46	2.93	0.21
epistle	100	792.07	107.67	27.75	2.54	0.3
lament	100	234.72	34.0	3.54	0.94	0.31
stanza	100	98.28	17.73	0.63	0.2	0.02
villanelle	100	155.75	20.47	4.22	0.09	2.19
ballad	100	650.28	98.01	9.55	1.98	1.71
hymn	100	445.35	58.63	9.39	2.06	0.63
monoku	100	10.89	1.74	0.02	0.0	0.0
dirge	100	220.39	31.36	4.14	1.06	0.39
elegy	100	433.65	55.32	9.01	2.42	0.16
verse	100	501.56	72.6	10.01	0.63	0.15
epitaph	100	118.26	16.81	3.17	0.53	0.1

Figure 6: Characteristics of the most popular poem types

only once which would result in no option for a neural net to learn what such words mean. However, the solution will eventually be adapted to load a larger dataset, and we will try to ensure that the time involved in training potentially larger data will have optimal time complexity [or at least not the most naive ;)].