Poetry Analysis and Generation
Machine Learning for Natural Language Processing 2022

Louis AMRANI
ENSAE Paris
louis.amrani@ensae.fr

Rémi VIDAL
ENSAE Paris
remi.vidal@ensae.fr

Abstract

In this paper we present our work on poem classification and generation using a large corpus of poems, sorted by theme. We study the corpus, and construct a deep learning model to classify the poems by theme. We obtain 54% accuracy on 3 types of poems. We also create a deep learning model for poem generation. We compare its performances with that of 2-gram and 3-gram models.

1 Problem Framing

Poetry data is a huge source of textual data. This NLP project aims to analyze a corpus of English poetry, create a topic classifier and generate poetic content according to a specific theme. The code can be found on Github[1]. The main file is the Colab Notebook [2] which contains all the data processing, from imports to poem generation.

2 Data Description

The dataset used in this project can be downloaded here. It is comprised of two folders, both containing subfolders of poems. These poems are categorized by the form (e.g. haiku, sonnet, etc.) or topic (love, nature, joy, peace, etc.). For our study, we only keep the topics. In the notebook `cleaning_data.ipynb`, we convert the subfolders into a dataframe. It contains the title, the author, the topic and the content of more than 14,000 poems.

3 Experiments Protocol

The "Data exploration" section of the main notebook shows that the data is of good quality, with almost no missing values and few duplicates. The poems are well distributed within the 144 themes. We quickly studied the distribution of authors and title/content length. A first tokenization indicates that the vocabulary size of the corpus is between 100,000 and 115,000 words, depending on the tokenizer. After checking that Zipf's law was verified for our poetry content (fig 1), we draw wordclouds by theme in order to catch the most relevant words inside a topic (fig 2).

The classification task is to predict a topic for a given poetic content. However, it's obvious that classifying a poem among more than a hundred themes is unfeasible. Besides, 100 poems for a single class is not enough for a model to learn properly. We therefore propose to gather similar topics inside three more general themes : love, nature and sadness (bad feelings actually). We take care that the classes are balanced. After tokenizing the poetry content, we randomly split the data in 2/3 train, 1/6 validation and 1/6 test. For the embedding technique, we import the pre-trained embeddings of FastText, an open-source library that provides English word vectors built from webcrawl and Wikipedia. Our model is a neural network with the embedding layer that turns the token indices into dense vectors, one hidden layer, and a classification layer to compute class probabilities using softmax function. We add a 20% dropout strategy in order to prevent overfitting. We run the training on 30 epochs with a batch size of 64 and learning rate of 10^{-3}.

The generation task consisted in generating text with models trained on the poems dataset. We used an LSTM model with an embedding layer. We also made some experiments with pre-trained embeddings, but found that the model converged more slowly during training. The model was trained to predict the next word using n-grams inputs. Once the model was trained, it could be used to generate text by taking some words as input, then taking its outputs as input, as presented dur-

[1] Link to the Github repository.
[2] Link to the Colab notebook.

ing the course.

4 Results

To evaluate our classification model, we need to make it guess the theme of a poem in the test set. The score of the model is then the accuracy of the predictions on the test set. With the above parameters, we achieve an accuracy of 54%, which is not bad for a 3-classes classification with approximately 1500 data in each class (a random model would be at 33%).

	Precision	**Recall**	**F1-score**	**Support**
Love	0.47	0.51	0.49	183
Nature	0.62	0.59	0.60	323
Sadness	0.51	0.51	0.51	262
Accuracy : 0.54				768

Table 1: Results for the classification task

Two aspects make the classification difficult : on the one hand, and as we see in the notebook, some poems are affected to several themes in the database. On the second hand, even if it is affected to one topic, a poem can use several themes (for instance, the word "love" often appears in a nature poem, see fig 2). It shows that summarizing a poem with a unique theme can be quite restrictive and imprecise. Thus, even for a human, it could be harder than expected to classify a content in one of the three general themes constructed.

The generation model was trained on love poems. We used categorical crossentropy as loss function. After giving it a seed text, we get the following result :

my dearest love i love
you madly you dive ripe
lots blest adversity otter fail
you prologue sat repeatedly rejuvenate
entail ernestine htis births cooked
thirst oozing dappled shakes steel
steel asks dainty prologue dainty
' keep seats profound carriage
it difficulty juicy 1 juicy
bitterweet countless plaited concerto nivedita
weddingparty become roguish wondered

We were forced to remove the punctuation, otherwise it appeared too often in the generated text. Qualitatively, it has not so much sense.

For the baseline, we use the N-gram approach. This is a truncated count based language model that estimate probability of an upcoming sequence given the past. The probability of a sequence of k elements is, for $n = 3$ (trigrams) :

$$P(w_{1:k}) = P(w_1) \times P(w_2|w_1) \prod_{i=3}^{k} P(w_i|w_{i-2}, w_{i-1})$$

Trigrams for love poem is qualitatively better :

Loving's done with that last
lay. I'll have to
be I love you, because
you, one most master Kissing
slowly is better Far far
better than me. So much
love showered and so ultimately,
it's gonna be ?

To assess generation performance, we can use perplexity as a performance metric. A good model should give low perplexity score to valid English sentences and high score to invalid English sentences. For a given sentence s that contains n words $w_1, \ldots, w_n$, the perplexity of the sentence given the probability distribution P is :

$$\text{Perplexity}(s) = P(w_1, \ldots, w_n)^{-\frac{1}{n}}$$

For a bi-gram model we get for example :

Perplexity(**'i love you so much'**) : 29.8
Perplexity(**'your beautiful eyes.'**) : 45.2

However, this metric isn't generalizable, and only gives us a hint about the model's performances. We chose to not use it on the other models, and leave it to the reader to judge the performances of the models.

Finally, a quick word about complexity : the complexity of a neural network is quite complex to evaluate, but if we use the reasoning described here, for our classification for example, we would have this complexity with n=30, t = 3203, and 3 layers with input = 200, hidden = 300 and output = 3.

5 Conclusion and Discussion

The analysis of this poetic corpus made it possible to build two models : a theme classification model with more than 52% accuracy ; and a generation model, that generates poetry content according to a specific topic. The results of this last model are quite satisfactory with our resources. Of course they could have been improved by studying more data, and with greater technical resources to build more complex models.

Appendix

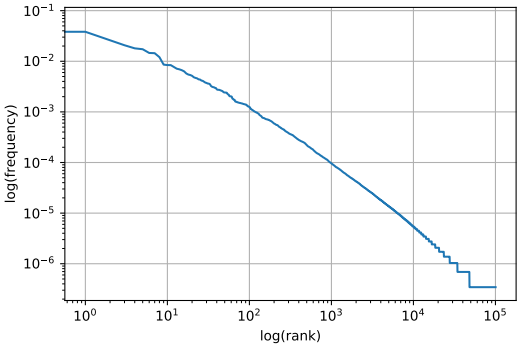

Figure 1: Log(rank) - Log(frequency) distribution using the TweetTokenizer

The plot is quite linear : Zipf's law is verified.

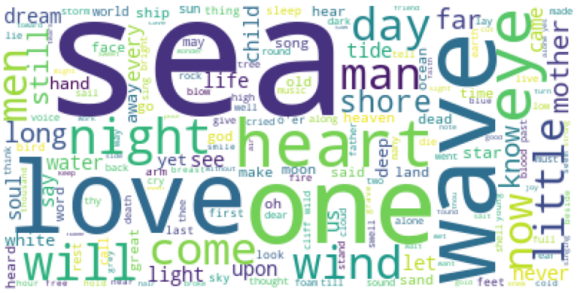

Figure 2: Wordcloud for the theme "sea"