A Chinese Person Name Generator: Based on MLP

Sijie Ju

1. Introduction

In this exercise, I trained a character-based language model to generate Chinese person names. To make the model work well, I converted the Chinese character into pinyin and conducted the training based on the pinyin representation. The training of this model was based on MLP neural network, and its quality was assessed using Negative Log-Likelihood, and also by manual evaluation.

2. Data and methods

The data I used for this model was downloaded from a Chinese name corpus (wainshine, 2022) in Github. This txt file has totally 1.2 million Chinese names which are composed of both first names and last names. The length of these names is either 2 or 3 characters. In Chinese, the last name is usually one character, and the first name is usually one or two characters. Despite the fact that in Chinese there exist last names composed of two characters, in the present data all the last names are only one-character. To obtain the single names, the data was split by line break and organized into a list.

Unlike alphabetic languages of which characters correspond to phonemes, Chinese characters correspond to morphemes, so the number of characters is enormous. There are totally 2270 characters used to form the names in the data. Compared to the 26-character vocabulary in English and the 33-character vocabulary in Russian, the vocabulary of Chinese is far more than these alphabetic languages. Due to the huge vocabulary, the training of a Chinese name generator is a time-consuming process. With a vocabulary of 2270 characters and one end marker, it requires around three hours to train the model.

To solve this problem, I used a library 'pypinyin' to convert the names into the form of pinyin, which marks the pronunciation of character, and built the vocabulary based on pinyin. By doing this, the vocabulary reduced from 2271 to 353. A dictionary from pinyin to characters was also built to convert the pinyin back to Chinese characters in the output. This method makes sense in this case because the ways of pronouncing a Chinese character are limited, one pinyin corresponds to various characters. What's more, there is no certain rule in the composition of Chinese names except that the last name should be chosen from a specific list of characters. To limit the conversion of the pinyin of last name, I built an extra dictionary for last names so that the characters for last name can be chosen from a correct list.

After building the vocabulary, I divided the names with the ratio of 8:1:1, creating the training dataset, evaluation dataset, and test dataset accordingly. Following this, I set the parameters of the model and started training with 200,000 iterations. The original output of the model were combinations of pinyins rather than characters. To address this, I added an additional step where a character would be randomly chosen from the list corresponding to the pinyin. For the assessment of model, I used the Negative Log-Likelihood as the loss function. My manual evaluation which is based on the output will be presented in the result section, providing qualitative insights into the model’s performance.

3. Result

By converting the characters into pinyin, the training time reduced to an average of 30 minutes. The training efficiency is greatly improved.

The test loss on the original parameters was 37.4112. After training, the best test loss that this model achieve is 3.6392, with the batch size of 64 and the starting learning rate of 0.1.

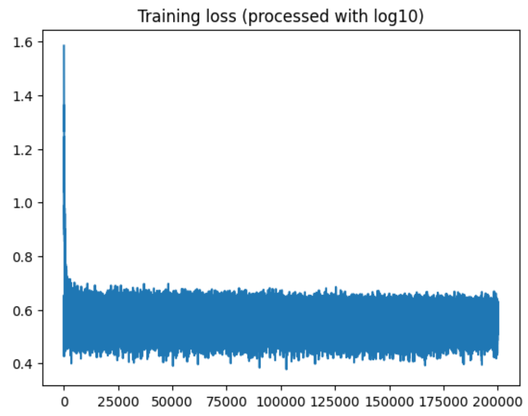

Figure 1 Training loss of the model (processed with $\log_{10}$)

Other sets of hyperparameters have also been tried, but the influence of both batch size and learning rate on the test loss is not obvious. The test loss fluctuated between 3.63 and 3.69. Although the loss value is not low enough, but it has reduced significantly from the original value.

Here I list twenty names that are generated once by this model.

(1) 飞问介，留滑董，归妲勤，叶挚，和骊浚，罗俊佑，申尖佶，纪骥君，连惟命，伏光，招质娜，兆威玠，王渭璨，章蠡，禄鸣危，乌埠乐，玉霈雄，彭号相，陈绣珺,宋润

From the perspective of a native Chinese speaker, the outputs of the model appear to be reasonable. First, there is no overgeneration of characters. The generated names follow the pattern of being either two or three characters in length. What’s more, since the characters used in these names comes from real person names, most of them resemble authentic names.

Although the output of this model looks well from the perspective of human, the current method actually has a huge problem. The process of converting the pinyin output to character relies on a random choice from the characters list corresponding to the given pinyin. It worked due to the lack of rules in the composition of Chinese names, but a more scientific and decent way is to train another model for the transformation from pinyin to characters.

Moreover, there is limitation in the current data. This corpus indeed provides abundant name data for training, but previously mentioned, the names in this corpus doesn’t have two-character last name. This limitation avoids the problem of generating non-existent last names, but also leads to the problem of monotony in the output.

4. Reference

Wainshine. (2022). *Chinese Names Corpus*. https://github.com/wainshine/Chinese-Names-Corpus/blob/master/Chinese_Names_Corpus/Chinese_Names_Corpus(120W).txt

Karpathy, A. (2022). *Makemore part 2 MLP*. https://github.com/karpathy/nn-zero-to-hero/blob/master/lectures/makemore/makemore_part2_mlp.ipynb