USE MCMC TO DO CHINESE STORY GENERATION

A PREPRINT

Shan Zhong[*]
Department of Statistics
University of Douth Carolina
Columbia, SC 29208
zhongs@email.sc.edu

2019 年 12 月 6 日

ABSTRACT

When I was young, I want to be a writer, but later I found math is more interesting and ended up as a STAT major student. Nevertheless, Automated story generation is always my dream. Here this project focus on the use of MCMC methods and Recurrent Neural Network to generate novel stories. From a favoriate author I like, I showed the technique to generate Chinese novel stories by sampling using conditional probabilities and acceptance criteria.

***K*eywords** Recurrent Neural Network · Bayesian · MCMC

1 introduction

This is a naive project to implement MCMC algorithm in a real world Chinese novel dataset. The methods used in this project are word2vec embedding, dynamic programming to do Chinese characters segmentation, recurrent neural network on subject classification, and finally MCMC algorithm to generate Chinese novel.

2 Tokens

One thing Chinese characters are different from English is that between Chinese characters, there are no spaces. We can not easily seperate words by space. Thus I utlized a Jieba Dictionary that use dynamic programming to find the most probable combination of characters.[3] Jieba is a package developed to cut the sentence into a more accurate segmentations, then make words suitable for text analysis. Example :

我们去吃饭. we go eat . Where 我们 means we, 去 means go, and 吃饭 means eat. The tokens are then "我们", "去" and " 吃饭".

[*] Shan is David's student)—*not* for acknowledging funding agencies.

3 Bigram

In a bigram setting, we can assume the distribution of tokens is conditional on the token previous to it(Naive idea, but for simplicity it is ok for an initial model). Thus the tokens becomes a Markov Chain.

P(我们去吃饭)=P(我们)P(去|我们)P(吃饭|我们去) ≈ P(我们)P(去|我们)P(吃饭|去)

There are also event representative models Martin(2017) use events to represent sentence by (subject , verb , object, modifier), every sentence in a story is reduced to a 4-word sentence. Sentence can be broken into mutiple events. [2] However, for the simplicity of the simulation, we just sample on the words directly.

4 MCMC

After we sample a sentence, how do we know the accuracy? Here I first do clustering on the original novel story, then I run the fitted clustering on the generated text to decide whether I should keep the text or not. Then in this way, I hope the automatically generated story can be stick to one specific topic.

5 Word vectors

When we want to classify texts, we need some sort of similarity between those word tokens. Then words with the similar meaning can be clustered. How do we represent the meaning of words? One solution is to use WordNet(a lexical database for the English language), WordNet use human labors to find synoym sets. Then we can know "good" is similar to "nice". But in this way, we cannot calculate a numerical similarity for those words.

The other way is to use word vectors. One way to do so is use one hot encoding to encode words as vectors, then the length of the vector is the size of the vocabulary. For example, the word "you" , "me" and "him" can then be represented as

$$you = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} me = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} him = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

A better way to do so is to represent words in a fixed dimension. For example, we can have a dimension to represent tense , a dimension to represent plural, and another dimension to represent negate prefix, etc.... We then let machine to learn the vectors itself.

How does the prediction works? There is the fancy idea "A word’ s meaning is given by the words that frequently appear close-by" ,we can build a model to predict occurrence of a word given another word is next to

表 1: Data summary

Original novel for sotry generation		
Name	Description	Size (counted Numbers)
Total Word	Total number of characters in the novel	7,927,322
Total Token	Total number of tokens in the novel	4,509,402
Unique Token	Unique number of tokens in the novel, the one we use for sample	80,046
Unique Bigram Token	Unique number of bigram tokens in the novel,	1,032,157

it. We can then use the posterior distribution to sample tokens and generate stories. Here I will try to implement the methods in Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean in their Efficient Estimation of Word Representations in Vector Space [1]. They represent each token by two vectors, one vector using the word surround to predict the token itself, one vector using itself to predict the word surround it.

6 pretrained embedding and Recurrent Neural Network

For recognizing the generated text and thus decide the rejection probabilities, we utilized a pre trained layer of Chinese embedding to max performance and simplify problem. The second layer is a stack of two LSTM layer to decide the subject the whole paragraph is talking about. We set a probability line. If the fitted accuracy of generated story is below a threshold. Then the story will go several steps back to regenerate.

Implementation I choiced my favorite novel "凡人修仙传" . I first seperate the whole novel by paragraphs. The story is of 7,927,322 words long. All the randomly generated story will be come from tokens from this novel, so the stories will be of similar style. The detail of the implenebtation is also attached . The codes use python.

参考文献

[1] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean Efficient Estimation of Word Representations in Vector Space In arXiv:1301.3781.

[2] Martin, L. J.; Ammanabrolu, P.; Hancock, W.; Singh, S.; Harrison, B.; and Riedl, M. O. Event representations for automated story generation with deep neural nets In arXiv:1706.01331.

[3] "Jieba" (Chinese for "to stutter") Chinese text segmentation: built to be the best Python Chinese word segmentation module.

[4] Guy Hadash, Einat Kermany, Boaz Carmeli, Ofer Lavi, George Kour, and Alon Jacovi. Estimate and replace: A novel approach to integrating deep neural networks with existing applications. *arXiv preprint arXiv:1804.09028*, 2018.