CHINESE STORY GENERATION

A PREPRINT

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

JianJun Hu
Department of Computer Science
University of South Carolina
Columbia, SC 29208
fake@stat.sc.edu

2019 年 12 月 12 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 techniques to generate Chinese novel stories by sampling using conditional probabilities and Recurrent Neural Network.

***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.[4] 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 Initial Bigram Model

How does the prediction work? There is the fancy idea "A word’s meaning is given by the words that frequently appear close-by" . In a bigram setting, we can assume the distribution of tokens is conditional on the token previous to it. 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. [1] However, for the simplicity of the simulation, we just sample on the words directly.

Implementation I choiced my favorite novel "凡人修仙传" for illustrating. 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. In the next few chapters, I used different technique to generate stories.

表 1: Data summary

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

表 2: Token summary

Description	Size (counted Numbers)
Unique tokens before ignoring	80,046
Ignoring tokens with frequency	< 5
Unique tokens after ignoring	25861

Bigram with firect count sampling Here are some generated sentence:

一只灵兽袋，极其可怕的从扇面上狂闪几下，忽然另外一眼，但这元婴虽然他不再流血。但是她一张惊艳之色。若是个清清楚楚，只有拳头直接镇压了灰烬。

For the direct count of "一只", we have 903 candidates, using bigram model, we just generate with probability equal to the observed frequency. We can clearly see a sparsity problem, as most of the counts are 1.

another sentence:

表 3: Direct count for word after '一只'

'手'	'手掌'	'拳头'	'手上'	'手臂'	'黄色'	'黑色'	'灰色',	','	'枯',	'小鸟',	'灵兽'	...
104	194	38	15	48	18	44	7	37	1	1	39	...

在韩立见过妖化之事一般，你，青光闪动一下从袖口中出价的住处后，晚辈这次下界来。风希心中更是一下，看样子并开始全心挑选并恰好从此无法。眼看一时欣喜的。白袍女子沉默了

Here we see the story we generated using direct counts is incoherent and not that meanful in the whole structure. This model is just for fun, let's try with some more sophisticated model.

4 Word vectors

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 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.

Word2vec model 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 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 [2]. 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.

Word2vec model use one vector to predict the words around the center, another vector to predict the center from the words around it. When we want to estimate P(Around|Center), we dot product these two vector, to get their similarity. Then these vectors can be learned to maximize the likelihood.

$$minimize\ J = -log\ P(w_{c-m}\ldots,w_{c-1},w_{c+1},\ldots,w_{c+m}|w_c) \tag{1}$$

$$= -log \coprod_{j=0,j\neq m}^{2m} P(w_{c-m+j}|w_c) \tag{2}$$

$$= -log \coprod_{j=0,j\neq m}^{2m} \frac{exp(u_{c-m+j}^T v_c)}{\sum_{k=1}^{|V|} exp(u_k^T v_c)} \tag{3}$$

$$= -\sum_{j=0,j\neq m} u_{c-m+j}^T v_c + 2m\ log\Big(\sum_{k+1}^{|V|} exp(u_k^T v_c)\Big) \tag{4}$$

$$U_{around} = \begin{pmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{pmatrix} \quad v_{center} = \begin{pmatrix} \square & \square & \square & \square \\ [\square & \square & \square & \square] \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{pmatrix} \quad U_{around} \cdot v_{center}[k,:]^T = \begin{bmatrix} \square \\ \square \\ \square \\ \square \\ \square \\ \square \end{bmatrix}$$

$$P = Softmax(U_{around} \cdot v_{center}[k,:]^T) = \coprod_{j=0,j\neq m}^{2m} \frac{exp(u_{c-m+j}^T v_c)}{\sum_{k=1}^{|V|} exp(u_k^T v_c)} = \begin{bmatrix} \square \\ \square \\ \square \\ \square \\ \square \\ \square \\ \square \end{bmatrix}$$

Negative Sampling It is computational expense to calculate |V| sum and update using the above formula, so instead we sample a part from the vocabulary with a noice distribution that give unrelated tokens less chance to be in the sample.[3]. This can be achieved by introducing a variable D to indicate whether the paired tokens are in the corpus.Then the equation becomes:

$$minimize\ J = argmax \coprod_{(w,c)\in D} P(D\ in\ corpus|w,c) \coprod_{(w,c)\in D_{Negative}} P(D\ not\ in\ corpus|w,c) \tag{5}$$

$$= -\sum_{(w,c)\in D} log\Big(\frac{1}{1+exp(-u_w^T v_c}\Big) - \sum_{(w,c)\in D_{Negative}} log\Big(\frac{1}{1+exp(u_w^T v_c}\Big) \tag{6}$$

Fixed window neural language model

5 Recurrent Neural Network

One thing we want to improve is that there might be useful information outside the window given. A RNN is trained by fitting the next word using all previous tokens. This is done by adding a hidden states layer. Each time, we use the previous state h_{t-1} and the token x_t to predict h_t. Then we predict $\hat{y}_t$ based on the states at h_t

$$h_t = \sigma(W^{(hh)}h_{t-1} + W^{(hx)}X_{[t]}) \tag{7}$$

$$\hat{y}_t = softmax(W^{(S)}h_t) \tag{8}$$

Here are some generated sentence, based on characters as input x_t, after 40,000 trails of training:

一些古怪的收进了一丝而心。只见此海面带瞬间会一路帮下，但只是很坐天雷本来到里，才还是夫人才能不想，还有些喜色惧色。第三卷风，一点切靠的，木女，有点点头一张不仅的。因为满发着他神则有本不用一劫，有不妙带着二人，妾测！韩立老在了几声不见臂，从储物袋中解神极的解目，绝不是对手的端到了。现到请我？他一人要是天星阵重炼之人的话，面镇清楚没有什么长丝毫不妙的，根本没有为了加笑，望了自然不假的神神目一丝天天而急

Problem with Token level RNN model When I use an token level RNN model with one hot encoding

6 Pretrained embedding

There are great similarity for word embedding with the same language. So on token level, we can utilize a pre trained layer of Chinese embedding to improve performance and simplify problem. [5]. I used a pretrained 300 dimension word2vec embeddings layer to represent correlation between tokens. These vectors were trained on a mixed corpus from Baidu Encyclopedia, Wikipedia zh, People's Daily News , Sogou News, Financial News, Zhihu QA, and Weibo，

表 4: Training Corpus Data summary

Corpus	Size	Tokens	Vocabulary Size
Mixed-large	22.6G	4037M	10653K

表 5: Training Setting

Window Size	Negative Sampling	Dimension	Low-Frequency Word
5	5	300	10

In the table above, Window Size is the steps look forward and backward, Negative sampling is the number of negative samples taken, Low-Frequency Word is the threshold for filter valid tokens.

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.

7 What to do next

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

参考文献

[1] 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.

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

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean Distributed Representations of Words and Phrases and their Compositionality In arXiv:1310.4546v1

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

[5] Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, Xiaoyong Du Analogical Reasoning on Chinese Morphological and Semantic Relations In ACL 2018.

[6] 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.