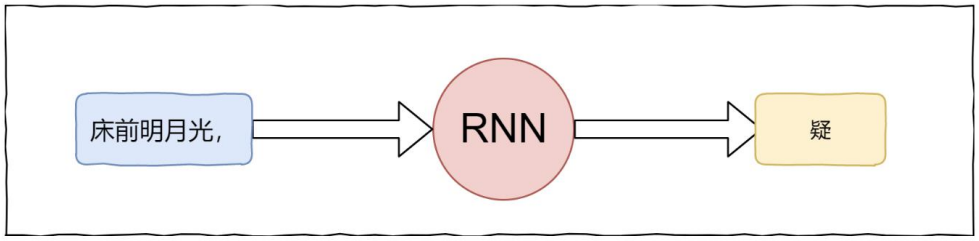
RNN_Poetry_Generator

1. Abstract

This is a Chinese poem generator based on recurrent neural network. Firstly, the sample data set is obtained. Then the first sentence language model is used to select words according to the structure of the first sentence and generate the first sentence. After the first sentence is obtained, the context model is used to compress the sentence vector, and the compressed vector is fed to the recursive neural network for training. The system uses the trained neural network to generate and identify poems automatically. Through the neural network learning of large-scale poetry samples, the characteristics of Chinese classical poetry creation are integrated into machine learning, which realizes the creation of Chinese classical poetry again, provides new ideas for the understanding of poetry, and provides inheritance and help for the majority of Chinese classical poetry.



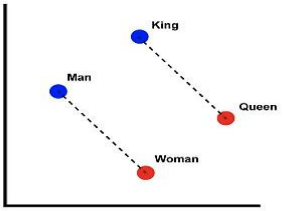
2. Introduction

Classical Chinese poetry is traditional Chinese poetry written in Classical Chinese and typified by certain traditional forms, or modes; traditional genres; and connections with particular historical periods, such as the poetry of the Tang dynasty. The existence of classical Chinese poetry is documented at least as early as the publication of the Classic of Poetry. Various combinations of forms and genres have developed over the ages. Many or most of these poetic forms were developed by the end of the Tang dynasty. Another key aspect of Classical Chinese poetry is its intense inter-relationship with other forms of Chinese art, such as Chinese painting and Chinese calligraphy. Classical Chinese poetry has proven to be of immense influence upon poetry worldwide.

2.1 The word vector

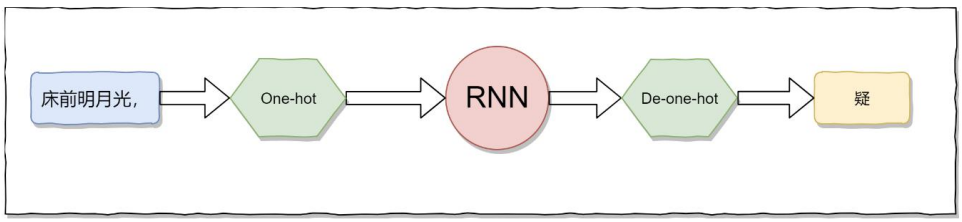
Word vectors represent a significant leap forward in advancing our ability to analyze relationships across words, sentences, and documents. In doing so, they advance technology by providing machines with much more information about words than has previously been possible using traditional representations of words. It is word vectors that make technologies such as speech recognition and machine translation possible. There are many excellent explanations of word vectors, but in this one, I want to

make the concept accessible to data and research people who aren't very familiar with natural language processing (NLP). At the level of the actual code used to generate word vectors, what we have been calling a word is simply a string — a series of characters with no particular representational content except to us as humans. The goal of word vectors is to model the representational content we humans have assigned to that string so that it can become a word not just for us but for the computer too.

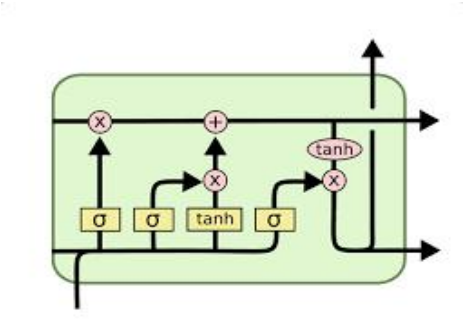


2.2 RNN

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes. This allows it to exhibit temporal dynamic behavior. Derived from feed forward neural networks, RNN can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. Recurrent neural networks are theoretically Turing complete and can run arbitrary programs to process arbitrary sequences of inputs.



LSTM is a very popular cyclic neural network. Compared with simple RNN, it has a poor understanding ability for long sentences (because RNN updates network state indiscriminately, resulting in the loss of the above information). However, LSTM updates network state selectively by adding four logic gates. This makes LSTM more effective for inference of long text semantics.



3. Implementation

Dataset processing

We used the text of 40,000 Tang poems as a training set, read the text, and divided it by line to form a poem list. Considering the size of the model, we only reserved poems with content less than a certain length.

There is a class in the code, tokenizer, which is a helper class written to facilitate the operation of character to numbering, number to character, string to numbering sequence, numbering sequence to string, and so on.

> 度秋:夏律昨留灰，秋箭今移晷。峨嵋岫初出，洞庭波渐起。桂白发幽岩，菊黄开灞涘。运流方可叹，含毫属微理。
> 仪鸾殿早秋:寒惊蓟门叶，秋发小山枝。松阴背日转，竹影避风移。提壶菊花岸，高兴芙蓉池。欲知凉气早，巢空燕不窥。
> 秋日即目:爽气浮丹阙，秋光澹紫宫。衣碎荷疏影，花明菊点丛。袍轻低草露，盖侧舞松风。散岫飘云叶，迷路飞烟鸿。砌
> 山阁晚秋:山亭秋色满，岩牖凉风度。疏兰尚染烟，残菊犹承露。古石衣新苔，新巢封古树。历览情无极，咫尺轮光暮。

Train model

A sequential model was built using tf.keras.Sequential, with Adam selected as the optimizer and cross entropy as the loss function. Two methods, one for random generation of ancient poetry, the other for random generation of collage poetry. In the training, we only use the method of randomly generating ancient poems to observe the effect.

The model can be trained by executing the train.py script. At the beginning of training, the output of the model is a little messy. Words and punctuation are jumbled together with no form at all. Starting with the second and third epochs, the output of the model is somewhat formatted. But there are some obvious problems, such as mispunctuation and different sentence lengths.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 128)         439552
_________________________________________________________________
lstm (LSTM)                  (None, None, 128)         131584
_________________________________________________________________
lstm_1 (LSTM)                (None, None, 128)         131584
_________________________________________________________________
time_distributed (TimeDistri (None, None, 3434)        442986
=================================================================
Total params: 1,145,706
Trainable params: 1,145,706
Non-trainable params: 0
```

Output

Load the trained model and generate the Chinese poem and the collage poem through the tool method.

江郭茫茫径复霜，月明帆落半相思。莫伤白日犹深笑，明月同来见五京。
床前明月光，水在紫楼长。水晓风犹尽，山从雪暮霜。风枝回枕急，湖树向流塘。
早日知同客，沧溟满谷川。
海上长初入，阔山谁独同。天开天际里，空尽一声声。

4. Conclusion

Through the study of RNN, it can automatically generate the bottom link according to the top link, write a complete poem according to the first sentence of the poem provided, or automatically generate a hidden poem according to the sentence given by the user. This is an LSTM model about RNN, its core is to use LSTM to predict the next word and achieve the purpose of text generation. In practice, the model does learn a few things, such as how to use punctuation to split sentences, how to organize phrases, and how to distribute subject-verb-object sentences. It can be seen from the model verification results that the trained network model can automatically generate relatively neat bottom link according to the top link. If the number of training iterations can be continued to increase, more excellent results will be obtained. In terms of poetry, the generated poems are also more suitable for the first sentence given by users, and the model also perfectly realizes the automatic generation of the hidden poems with excellent results.

Reference

1. Poetry Generator: blog.csdn.net/aaronjny/article/details/79677457
2. RNN: wikipedia.org/wiki/Recurrent_neural_network
3. Word Vector: wikipedia.org/wiki/Word2vec
4. LSTM: en.wikipedia.org/wiki/Long_short-term_memory