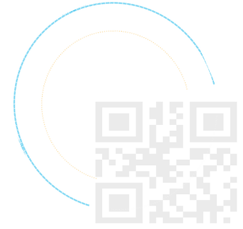
文本自动生成—实践

玖强



Outline

- 文本到文本生成
- 图像到文本生成

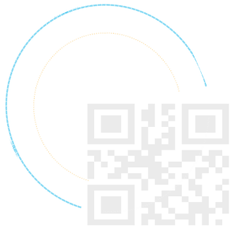
文本到文本生成

白鹭窥鱼立, Egrets stood, peeping fishes. 青山照水开. Water was still, reflecting mountains. 夜来风不动, The wind went down by nightfall, 明月见楼台. as the moon came up by the tower.	满怀风月一枝春, Budding branches are full of romance. 未见梅花亦可人. Plum blossoms are invisible but adorable. 不为东风无此客, With the east wind comes Spring. 世间何处是前身. Where on earth do I come from?

Chinese Poetry Generation with RNNs

Zhang, Xingxing, and Mirella Lapata. "Chinese poetry generation with recurrent neural networks." Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014.

- 基于RNN的中国诗歌生成模型
- 生成模型通过学习到单个字符和由单个字符结合而成的诗句的形式，以及它们之间是如何相互强化和限制，来进行内容提取和结果呈现的工作



相关工作

- 和图像到文本生成一样：绝大多数之前的方法是采用了模板的方式，根据一系列限制，结合基于语料库和字典式资源，来生成文本。
- 2008年Tosa等人和2009年Wu等人的俳句生成器模型就是根据从语料库和额外的词汇资源提取出的规则来扩大用户的查询需求

Naoko Tosa, Hideto Obara, and Michihiko Minoh.2008. Hitch Haiku: An Interactive Supporting Sys-tem for Composing Haiku Poem How I Learned toLove the Bomb: Defcon and the Ethics of Com-puter Games.InProceedings of the 7th Inter-national Conference on Entertainment Computing,pages 209–216, Pittsburgh, PA.

- 2009年的Netzer等人通过已经建立的联想词汇库生成俳句等等

Yael Netzer, David Gabay, Yoav Goldberg, andMichael Elhadad. 2009. Gaiku: Generating Haikuwith Word Associations Norms. InProceedings ofthe Workshop on Computational Approaches to Lin-guistic Creativity, pages 32–39, Boulder, Colorado.

- 统计机器翻译和相关的文本生成应用启发后：

2010年Greene等人从一个他们后来使用的诗歌文本语料库以及加权有限状态转换器中推断出了诗的韵律；Xingxing zhang也提出了本次实验的rnn-based诗词模型，应该是第一次用到中文的。

诗歌结构–以四行诗为例

四行诗和律诗是中国古代诗歌中比较有名的部分，它们都必须满足诗歌的结构性、音调性和语义性的要求。
下面先看四行诗的例子：

相 思

Missing You

红 豆 生 南 国，（*Z P P Z）

Red berries born in the warm southland.

春 来 发 几 **枝**？（P P Z Z P）

How many branches flush in the spring?

愿 君 多 采 撷，（*P P Z Z）

Take home an armful, for my sake,

此 物 最 相 **思**。（*Z Z P P）

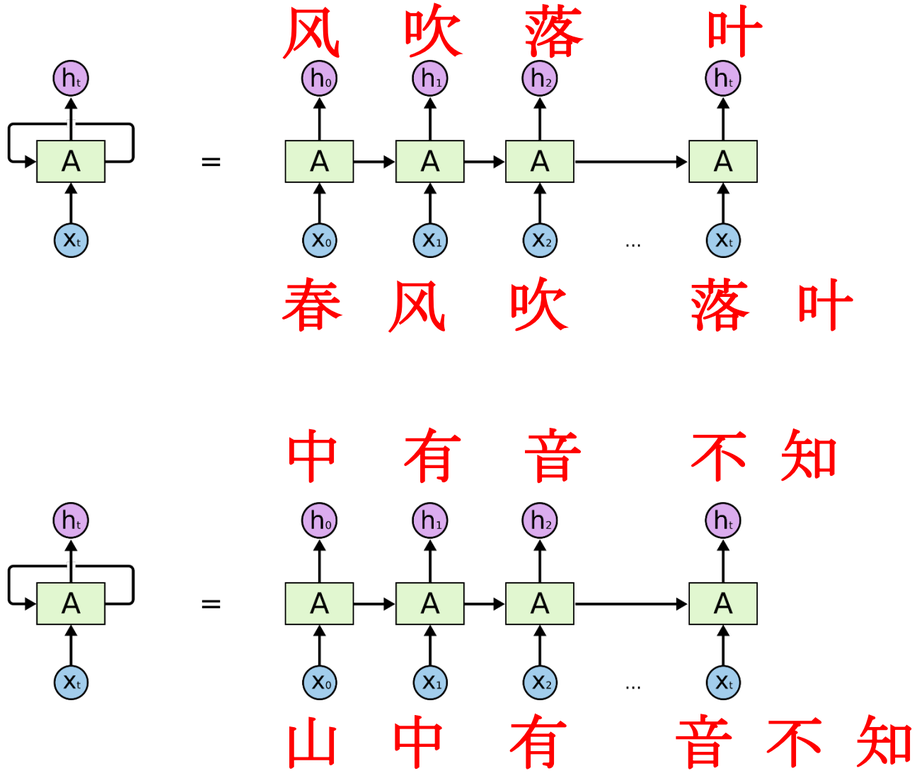
As a symbol of our love.

- 基本属性：

1. 有四行
2. 每一行有五个或七个字符，
3. 字符与字符间、行与行间都满足特定的音韵模式。
   - 例如：第二、四行和第一行的最后一个字符必须押韵，第三行没有限制。
   - 另外，诗歌必须遵循规定的声调模式，每个字符只有一种声调：平声或仄声。
   - 除此以外，诗歌对语言的运用必须准确无误，能让读者仿佛身临其境。

生成模型

- **recurrent generation model (RGM)**
- **RGM模型通过考虑RCM模型输出的向量和当前行已产生字符来得到下一个字符的概率分布虑**



生成模型

- **recurrent generation model (RGM)**
- **RGM模型通过考虑RCM模型输出的向量和当前行已产生字符来得到下一个字符的概率分布**

$S_{i+1} = w_1, w_2, \ldots, w_m$ 表示将要生成的行。RGM模型目的是要计算 $P(w_{j+1}|w_{1:j}, S_{1:i})$ ，因为前 i 行已经被编码为文本向量 u_i^j ，所以我们计算的是 $P(w_{j+1}|w_{1:j}, u_i^j)$ 。因此，

$$P(S_{i+1}|S_{1:i}) = \prod_{j=1}^{m-1} P(w_{j+1}|w_{1:j}, u_i^j)$$

$|V|$ 表示字符集的大小。矩阵 $H \in R^{q \times q}$ 将文本向量 u_i^j 转换为隐藏表示。矩阵 $X \in R^{q \times |V|}$ 将一个字符转换为隐藏表示。矩阵 $R \in R^{q \times q}$ 完成循环转换。矩阵 $Y \in R^{|V| \times q}$ 将隐藏表示解码为字符集中所有字符的权重。 w 表示字符集 V 中索引为 k 的字符。 r_j 是RGM模型在第 j 步的隐藏层。 y_{j+1} 是RGM模型在第 j 步的输出。RGM模型表示为：

$$r_0 = 0$$

$$r_j = \sigma(R \cdot r_{j-1} + X \cdot e(w_j) + H \cdot u_i^j)$$

$$y_{j+1} = Y \cdot r_j$$

所以，如果 σ 是softmax函数，则

$$P(w_{j+1} = k|w_{1:j}, u_i^j) = \frac{e^{y_{j+1,k}}}{\sum_{k=1}^{|V|} e^{y_{i+1,k}}}$$

模型定义

```python
import torch
import torch.nn as nn
from torch.autograd import Variable
import torch.nn.functional as F
# import nninit


class PoetryModel(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim):
        super(PoetryModel, self).__init__()
        self.hidden_dim = hidden_dim
        self.embeddings = nn.Embedding(vocab_size, embedding_dim)
        self.lstm = nn.LSTM(embedding_dim, self.hidden_dim)

        self.linear1 = nn.Linear(self.hidden_dim, vocab_size)
        # self.dropout = nn.Dropout(0.2)
        self.softmax = nn.LogSoftmax()

    def forward(self, input, hidden):
        length = input.size()[0]
        embeds = self.embeddings(input).view((length, 1, -1))
        output, hidden = self.lstm(embeds, hidden)
        output = F.relu(self.linear1(output.view(length, -1)))
        # output = self.dropout(output)
        output = self.softmax(output)
        return output, hidden

    def initHidden(self, length=1):
        return (Variable(torch.zeros(length, 1, self.hidden_dim).cuda()),
                Variable(torch.zeros(length, 1, self.hidden_dim)).cuda())
```

Word embedding define

Text generation model define

Word embedding forward

Text generation model forward

Decoding

The probability of the $(j+1)$th word given the previous j words and the previous i lines is estimated by a *softmax* function:

$$P(w_{j+1} = k | w_{1:j}, u_i^j) = \frac{\exp(y_{j+1,k})}{\sum_{k=1}^{|V|} \exp(y_{j+1,k})}$$

训练数据

https://github.com/chinese-poetry/chinese-poetry

- 中华古典文集数据库, 包含5.5万首唐诗、26万首宋诗和2.1万首宋词. 唐宋两朝近1.4万古诗人, 和两宋时期1.5K词人.
- 古诗数据分发采用繁体字的分组JSON文件, 保留繁体能更大程度地保存原数据.

古诗JSON结构

```
[
  {
    "strains": [
      "平平平仄仄，平仄仄平平。",
      "仄仄平平仄，平平仄仄平。",
      "平平平仄仄，平仄仄平平。",
      "平仄仄平仄，平平仄仄平。"
    ],
    "author": "太宗皇帝",
    "paragraphs": [
      "秦川雄帝宅，函谷壯皇居。",
      "綺殿千尋起，離宮百雉餘。",
      "連薨遙接漢，飛觀迴凌虛。",
      "雲日隱層闕，風煙出綺疎。"
    ],
    "title": "帝京篇十首 一"
  },
  ... 每单个JSON文件1000条唐诗记录.
]
```

```
import json
data = json.loads(open(file).read())
```

加载数据

```
data = []
src = './chinese-poetry/json/'
for filename in os.listdir(src):
    if filename.startswith("poet.tang"):
        data.extend(handleJson(src+filename))
return data
```

```
def handleJson(file):
    # print file
    rst = []
    data = json.loads(open(file).read())
    for poetry in data:
        pdata = ""
        if (author!=None and poetry.get("author")!=author):
            continue
        p = poetry.get("paragraphs")
        flag = False
        for s in p:
            sp = re.split("[, ! 。]".decode("utf-8"), s)
            for tr in sp:
                if constrain != None and len(tr) != constrain and len(tr)!=0:
                    flag = True
                    break
                if flag:
                    break
        if flag:
            continue
        for sentence in poetry.get("paragraphs"):
            pdata += sentence
        pdata = sentenceParse(pdata)
        if pdata!="":
            rst.append(pdata)
    return rst
```

数据预处理—生成字典与编码

字典生成

```
for s in data:
    print s
word_to_ix = {}

for sent in data:
    for word in sent:
        if word not in word_to_ix:
            word_to_ix[word] = len(word_to_ix)
word_to_ix['<EOP>'] = len(word_to_ix)
word_to_ix['<START>'] = len(word_to_ix)

VOCAB_SIZE = len(word_to_ix)

print "VOCAB_SIZE:", VOCAB_SIZE
```

输出one-hot编码

```
t, o = makeForOneCase(s, one_hot_var_target)
```

输入one-hot编码

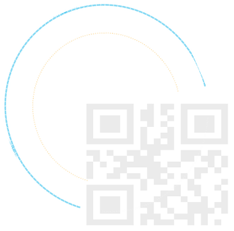
```
46  one_hot_var_target = {}
47  for w in word_to_ix:
48      one_hot_var_target.setdefault(w, make_one_hot_vec_target(w, word_to_ix))
```

功能函数

```
def make_one_hot_vec(word, word_to_ix):
    rst = torch.zeros(1, 1, len(word_to_ix))
    rst[0][0][word_to_ix[word]] = 1
    return autograd.Variable(rst)


def make_one_hot_vec_target(word, word_to_ix):
    rst = autograd.Variable(torch.LongTensor([word_to_ix[word]]))
    return rst


def makeForOneCase(s, one_hot_var_target):
    tmpIn = []
    tmpOut = []
    for i in range(1, len(s)):
        w = s[i]
        w_b = s[i - 1]
        tmpIn.append(one_hot_var_target[w_b])
        tmpOut.append(one_hot_var_target[w])
    return torch.cat(tmpIn), torch.cat(tmpOut)
```

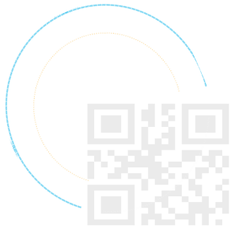
数据预处理—训练模型

```
for epoch in range(epochNum):
    for batchIndex in range(int(TRAINSIZE / batch)):
        model.zero_grad()
        loss = 0
        counts = 0
        for case in range(batchIndex * batch, min((batchIndex + 1) * batch, TRAINSIZE)):
            s = data[case]
            hidden = model.initHidden()
            t, o = makeForOneCase(s, one_hot_var_target)
            output, hidden = model(t.cuda(), hidden)
            loss += criterion(output, o.cuda())
            counts += 1
        loss = loss / counts
        loss.backward()
        print epoch, loss.data[0]
        optimizer.step()
    test()
torch.save(model, 'poetry-gen.pt')
```



数据预处理—测试程序

```
def test():
    v = int(TRAINSIZE / batch)
    loss = 0
    counts = 0
    for case in range(v * batch, min((v + 1) * batch, TRAINSIZE)):
        s = data[case]
        hidden = model.initHidden()
        t, o = makeForOneCase(s, one_hot_var_target)
        output, hidden = model(t.cuda(), hidden)
        loss += criterion(output, o.cuda())
        counts += 1
    loss = loss / counts
    print "=====",loss.data[0]
```



数据预处理—DEMO

```python
# Sample from a category and starting letter
def sample(startWord='<START>'):
    input = make_one_hot_vec_target(startWord, word_to_ix)
    hidden = model.initHidden()
    output_name = "";
    if (startWord != "<START>"):
        output_name = startWord
    for i in range(max_length):
        output, hidden = model(input.cuda(), hidden)
        topv, topi = output.data.topk(1)
        topi = topi[0][0]
        w = ix_to_word[topi]
        if w == "<EOP>":
            break
        else:
            output_name += w
        input = make_one_hot_vec_target(w, word_to_ix)
    return output_name
```

```
Traceback (most recent call last):
  File "sample.py", line 39, in <module>
    print sample("春".decode('utf-8'))
  File "sample.py", line 26, in sample
    output, hidden = model(input.cuda(), hidden)
  File "/usr/local/lib/python2.7/dist-packages/torch/nn/modules/module.py", line 325, in __call__
    result = self.forward(*input, **kwargs)
  File "/media/jxgu/github/NLP_Practice.PyTorch/chinese_poetry_gen/model.py", line 25, in forward
    output = self.softmax(output)
  File "/usr/local/lib/python2.7/dist-packages/torch/nn/modules/module.py", line 323, in __call__
    for hook in self._forward_pre_hooks.values():
  File "/usr/local/lib/python2.7/dist-packages/torch/nn/modules/module.py", line 366, in __getattr__
    type(self).__name__, name))
AttributeError: 'LogSoftmax' object has no attribute '_forward_pre_hooks'
```

Change your pytorch!

http://pytorch.org/previous-versions/

```python
print sample("春".decode('utf-8'))
print sample("花".decode('utf-8'))
print sample("秋".decode('utf-8'))
print sample("月".decode('utf-8'))
print sample("夜".decode('utf-8'))
print sample("山".decode('utf-8'))
print sample("水".decode('utf-8'))
print sample("葉".decode('utf-8'))
```

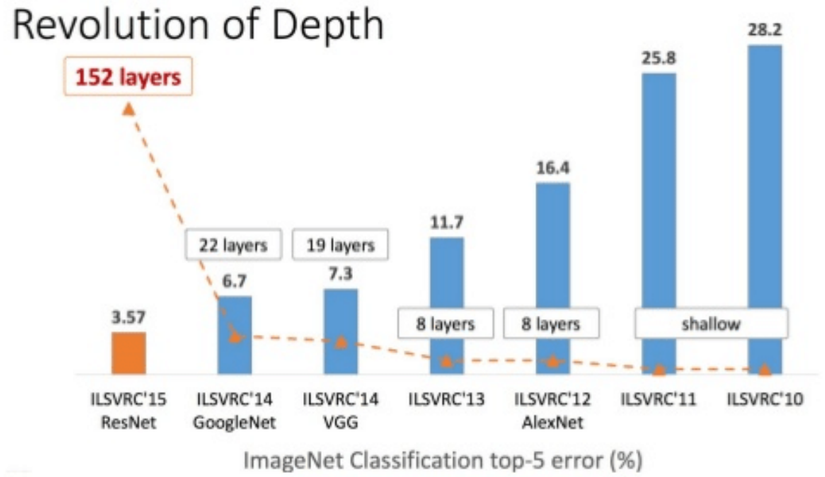
图像到文本生成介绍

Computer vision

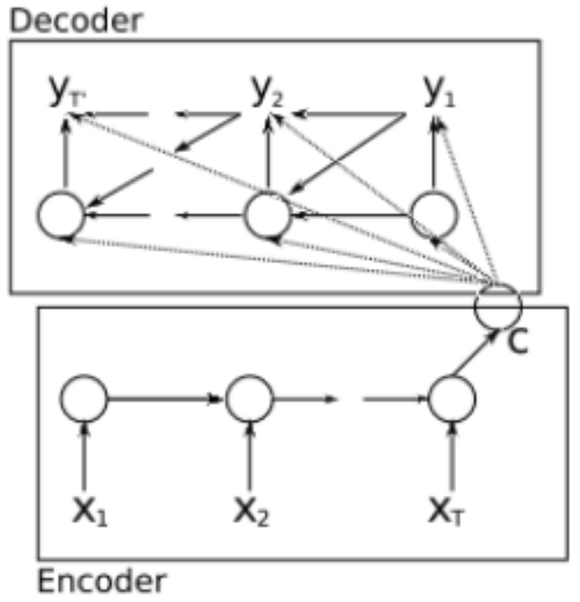
Natural language processing

背景回顾

1. Success in image classification/recognition
2. Close to human level performance
3. Deep CNN’s, Big Datasets
4. Image to fixed length vector



1. Machine Translation
2. Language generating RNN’s
3. Decoder-Encoder framework
4. Maximize likelihood of target sentence



Data Sets

Flickr8k

8000 images, each annotated with 5 sentences via AMT

1000 for validation, testing

Flickr 30k

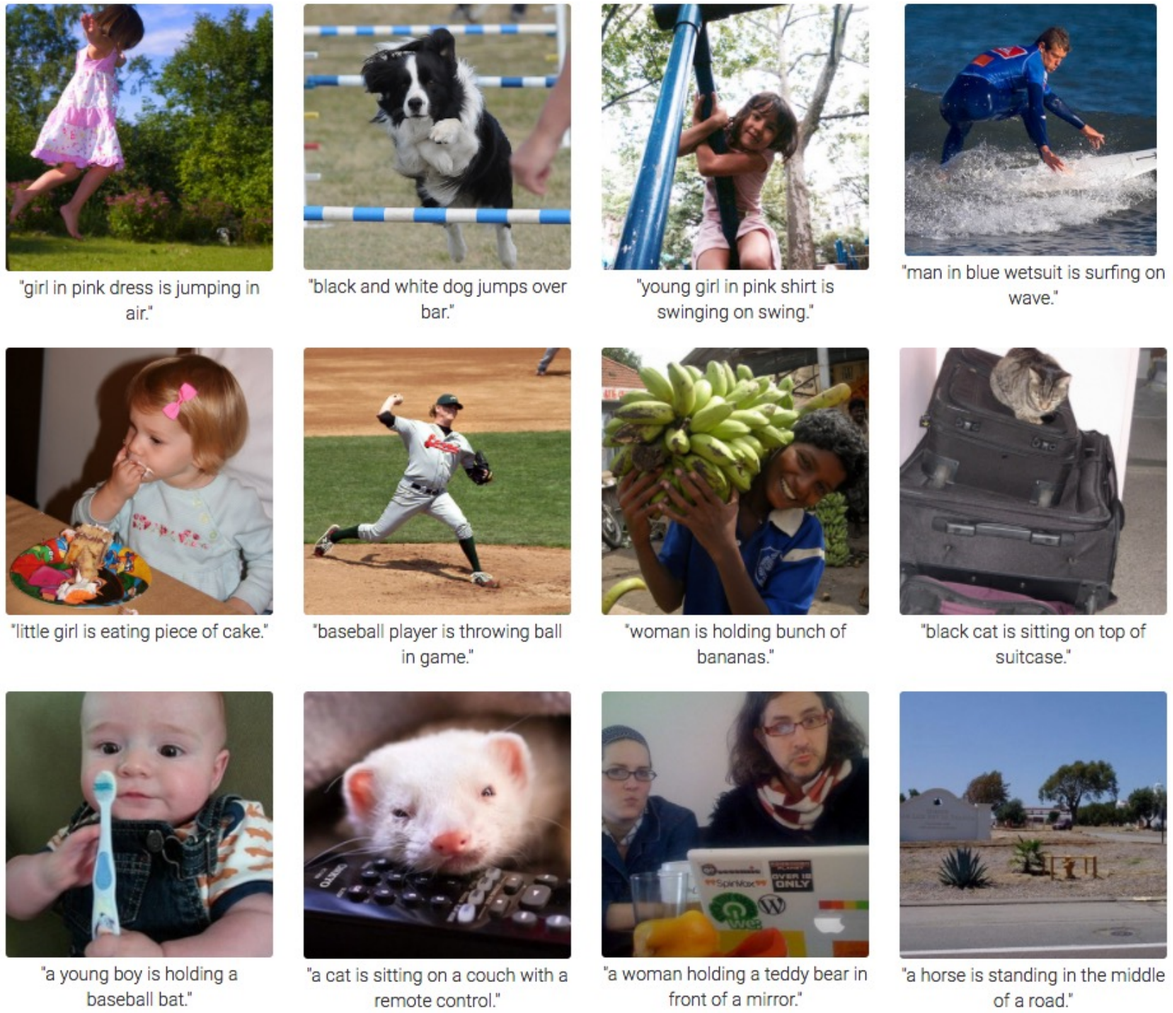
30k images

1000 validation, 1000 testing

MSCOCO

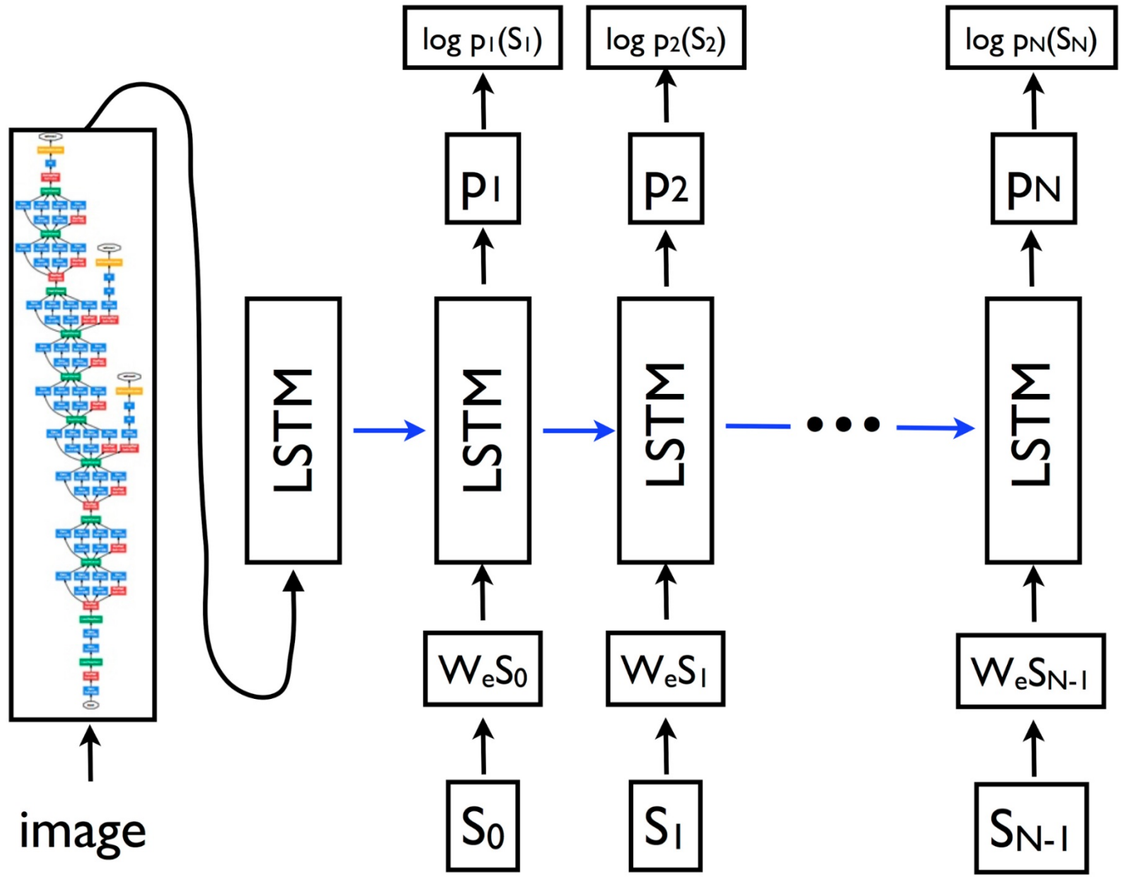
123,000 images

5000 for validation, testing



RNN-based im2text起源

1. Combine Vision CNN with Language RNN
2. Deep CNN as encoder
3. Language Generating RNN as decoder
4. End to end model
5. Maximize p(S|I)



Vinyals, Oriol, et al. "Show and tell: Lessons learned from the 2015 mscoco image captioning challenge." IEEE transactions on pattern analysis and machine intelligence 39.4 (2017): 652-663.

Representing Images

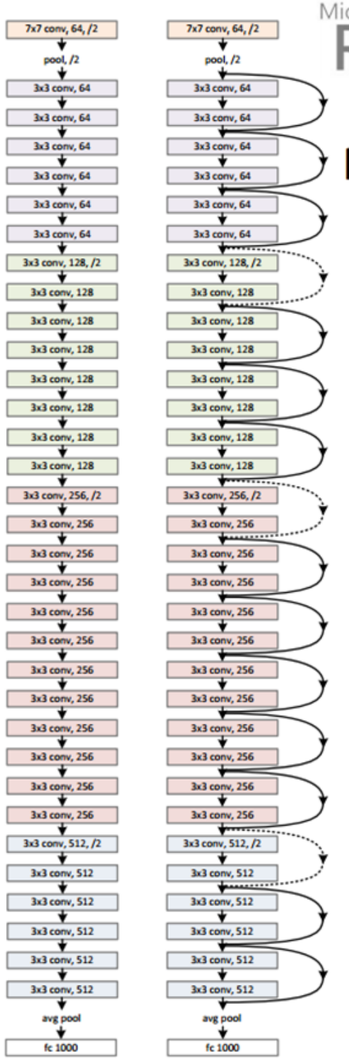
Process image with a “Convolutional Neural Net” pretrained on ImageNet

$$v = W_m[CNN_{\theta_c}(I_b)] + b_m$$

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

plain net

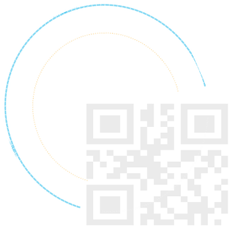
ResNet



Microsoft Research

Representing Images

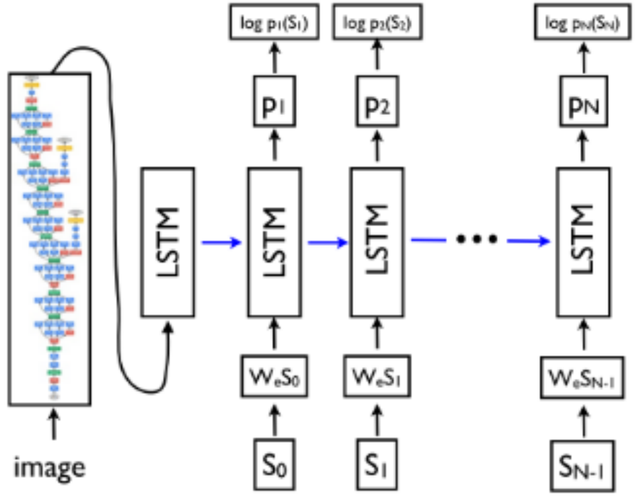
```
class myResnet(nn.Module):
    def __init__(self, resnet):
        super(myResnet, self).__init__()
        self.resnet = resnet

    def forward(self, img, att_size=14):
        x = img.unsqueeze(0)

        x = self.resnet.conv1(x)
        x = self.resnet.bn1(x)
        x = self.resnet.relu(x)
        x = self.resnet.maxpool(x)

        x = self.resnet.layer1(x)
        x = self.resnet.layer2(x)
        x = self.resnet.layer3(x)
        x = self.resnet.layer4(x)

        #fc = x.mean(2).mean(2)
        #att = spatialAdaAvgPool(x,att_size,att_size).squeeze().permute(1, 2, 0)
        fc = x.mean(2).mean(2).squeeze()
        att = F.adaptive_avg_pool2d(x,[att_size,att_size]).squeeze().permute(1, 2, 0)

        return fc, att
```



Show and tell 图像描述模型

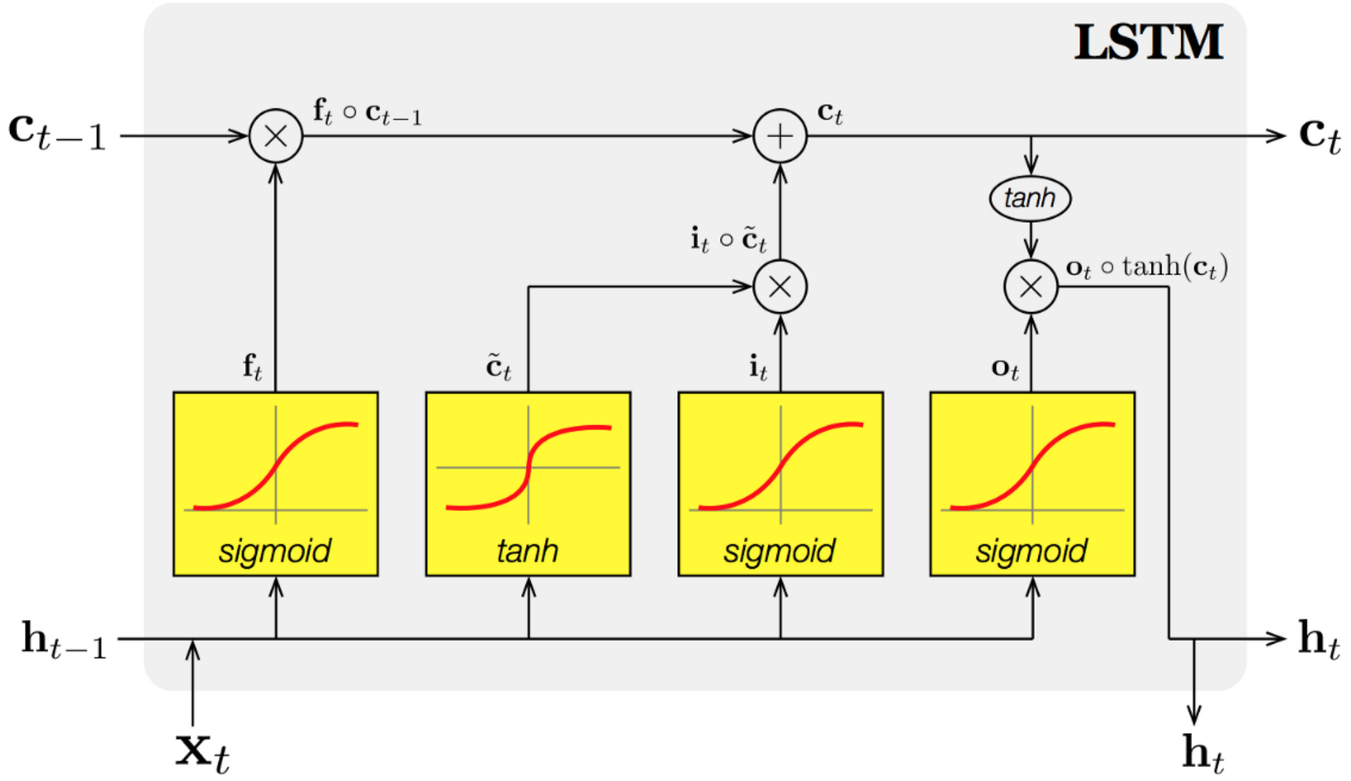
Neural Image Caption (NIC)

- CNN: 22 layer GoogleNet
- LSTM for modeling
  $\log p(S|I) = \sum_{t=0}^{N} \log p(S_t|I, S_0, \ldots, S_{t-1})$
- Word embedding W_e



LSTM-BASE 语言模型

- Predicts next word in sentence
- Memory cell for longer memory
- S_t one-hot vectors + START/END token
- $x_{-1} = \mathrm{CNN}(I)$, $x_t = W_e S_t$, $p_{t+1} = \mathrm{LSTM}(x_t)$



Gating variables

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_t\right)$$
$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i\right)$$
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o\right)$$

Candidate (memory) cell state

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c\right)$$

Cell & Hidden state

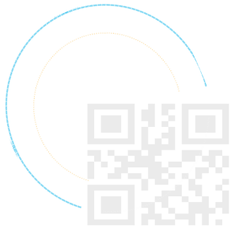
$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

Show and tell 图像描述模型

Branch: master ▾ Stack-Captioning / models / ShowTellModel.py

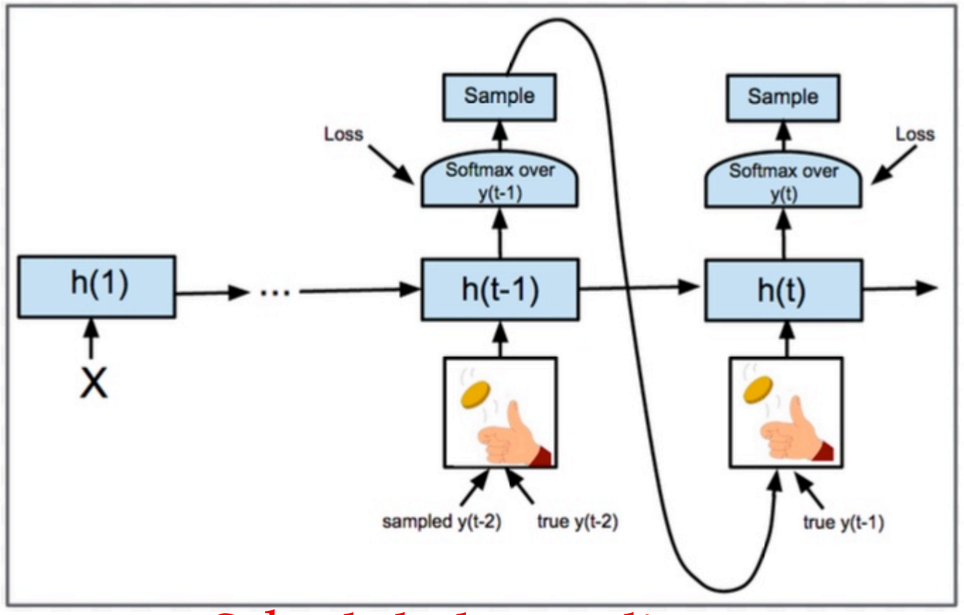
 gujiuxiang Initial commit

```
class ShowTellModel(nn.Module):
    def __init__(self, opt):
        super(ShowTellModel, self).__init__()
        self.vocab_size = opt.vocab_size
        self.input_encoding_size = opt.input_encoding_size
        self.rnn_type = opt.rnn_type
        self.rnn_size = opt.rnn_size
        self.num_layers = opt.num_layers
        self.drop_prob_lm = opt.drop_prob_lm
        self.seq_length = opt.seq_length
        self.fc_feat_size = opt.fc_feat_size

        self.ss_prob = 0.0 # Schedule sampling probability

        self.img_embed = nn.Linear(self.fc_feat_size, self.input_encoding_size)
        self.core = getattr(nn, self.rnn_type.upper())(self.input_encoding_size, self.rnn_size, self.num_layers, bias=False, dropout=self.c
        self.embed = nn.Embedding(self.vocab_size + 1, self.input_encoding_size)
        self.logit = nn.Linear(self.rnn_size, self.vocab_size + 1)
        self.dropout = nn.Dropout(self.drop_prob_lm)

        self.init_weights()
```



Training

- Loss function $\mathrm{L}(I, S) = -\sum_{t=1}^{N} \log p_t(S_t)$
- CNN pre-trained on ImageNet
- Minimize w.r.t. LSTM parameters, W_e and CNN top layer
- SGD on mini-batches
- Dropout and ensembling
- 512 dimensional embedding



Scheduled sampling

Training

```
def forward(self, fc_feats, att_feats, seq):
    batch_size = fc_feats.size(0)
    state = self.init_hidden(batch_size)
    outputs = []

    for i in range(seq.size(1)):
        if i == 0:
            xt = self.img_embed(fc_feats)
        else:
            if self.training and i >= 2 and self.ss_prob > 0.0: # otherwiste no need to sample
                sample_prob = fc_feats.data.new(batch_size).uniform_(0, 1)
                sample_mask = sample_prob < self.ss_prob
                if sample_mask.sum() == 0:
                    it = seq[:, i-1].clone()
                else:
                    sample_ind = sample_mask.nonzero().view(-1)
                    it = seq[:, i-1].data.clone()
                    #prob_prev = torch.exp(outputs[-1].data.index_select(0, sample_ind)) # fetch prev distribution: shape Nx(M+1)
                    #it.index_copy_(0, sample_ind, torch.multinomial(prob_prev, 1).view(-1))
                    prob_prev = torch.exp(outputs[-1].data) # fetch prev distribution: shape Nx(M+1)
                    it.index_copy_(0, sample_ind, torch.multinomial(prob_prev, 1).view(-1).index_select(0, sample_ind))
                    it = Variable(it, requires_grad=False)
            else:
                it = seq[:, i-1].clone()
            # break if all the sequences end
            if i >= 2 and seq[:, i-1].data.sum() == 0:
                break
            xt = self.embed(it)

        output, state = self.core(xt.unsqueeze(0), state)
        output = F.log_softmax(self.logit(self.dropout(output.squeeze(0))))
        outputs.append(output)

    return torch.cat([_.unsqueeze(1) for _ in outputs[1:]], 1).contiguous()
```

Generation

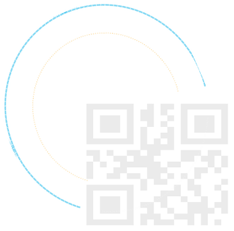
- Give $x_{-1} = \text{CNN}(I)$
- $x_0 = W_e S_0$, S_0 START token
- Sample word S_1
- Feed $W_e S_1$ to LSTM
- BeamSearch, beam size 20

Eval

The current code is a complete mess, I am too lazy to clean it up. If you run the two stage model, you will have the following results:

```
Beam size: 5, image 217951: a man is flying a kite in the water
Beam size: 5, image 130524: a desk with two laptops and a laptop computer
Beam size: 5, image 33759: a young boy swinging a baseball bat at a ball
Beam size: 5, image 281972: a young boy holding a baseball bat at a ball
Beam size: 5, image 321647: a baseball player holding a bat on a field
Beam size: 5, image 348877: a close up of a pizza on a table
Beam size: 5, image 504152: a kitchen with lots of tools hanging on a wall
Beam size: 5, image 335981: a group of people standing in front of a store
Beam size: 5, image 455974: an open refrigerator filled with lots of food
Beam size: 5, image 237501: two teddy bears sitting next to each other
Beam size: 5, image 572233: a bride and groom are cutting a wedding cake
Beam size: 5, image 560744: a man sitting at a table with a glass of wine
Beam size: 5, image 74478: a group of people standing around a table
evaluating validation preformance... -1/5000 (0.000000, with coarse_loss 0.000000)
coco-caption/annotations/captions_val2014.json
```

https://github.com/gujiuxiang/Stack-Captioning



Generation

```
def sample(self, fc_feats, att_feats, opt={}):
    sample_max = opt.get('sample_max', 1)
    beam_size = opt.get('beam_size', 1)
    temperature = opt.get('temperature', 1.0)
    if beam_size > 1:
        return self.sample_beam(fc_feats, att_feats, opt)

    batch_size = fc_feats.size(0)
    state = self.init_hidden(batch_size)
    seq = []
    seqLogprobs = []
    for t in range(self.seq_length + 2):
        if t == 0:
            xt = self.img_embed(fc_feats)
        else:
            if t == 1: # input <bos>
                it = fc_feats.data.new(batch_size).long().zero_()
            elif sample_max:
                sampleLogprobs, it = torch.max(logprobs.data, 1)
                it = it.view(-1).long()
            else:
                if temperature == 1.0:
                    prob_prev = torch.exp(logprobs.data).cpu() # fetch prev distribution: shape Nx(M+1)
                else:
                    # scale logprobs by temperature
                    prob_prev = torch.exp(torch.div(logprobs.data, temperature)).cpu()
                it = torch.multinomial(prob_prev, 1).cuda()
                sampleLogprobs = logprobs.gather(1, Variable(it, requires_grad=False)) # gather the logprobs at sampled positions
                it = it.view(-1).long() # and flatten indices for downstream processing

            xt = self.embed(Variable(it, requires_grad=False))

        if t >= 2:
            # stop when all finished
            if t == 2:
                unfinished = it > 0
            else:
                unfinished = unfinished * (it > 0)
            if unfinished.sum() == 0:
                break
            it = it * unfinished.type_as(it)
            seq.append(it) #seq[t] the input of t+2 time step
            seqLogprobs.append(sampleLogprobs.view(-1))

        output, state = self.core(xt.unsqueeze(0), state)
        logprobs = F.log_softmax(self.logit(self.dropout(output.squeeze(0))))

    return torch.cat([_.unsqueeze(1) for _ in seq], 1), torch.cat([_.unsqueeze(1) for _ in seqLogprobs], 1)
```

如果图像描述+诗歌生成=？

基本信息

编辑

静夜思

唐朝诗人 李白 字太白

床前明月光，

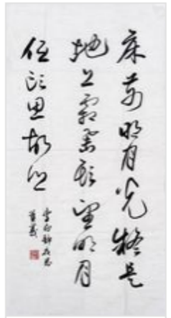
疑是地上霜。

举头望明月，

低头思故乡。

之所以强调版本，是因为唐朝版本《静夜思》是：床前看月光 [1] ，疑是地上霜。举头望山月，低头思故乡。

由于明清版本流传较广，这首《静夜思》中每一句都成为中国人心目中的强烈意象。伴随着众多的再创作，用诗句来作为作品题目的现象也就层出不穷。



董义书法作品

看图写诗？

END