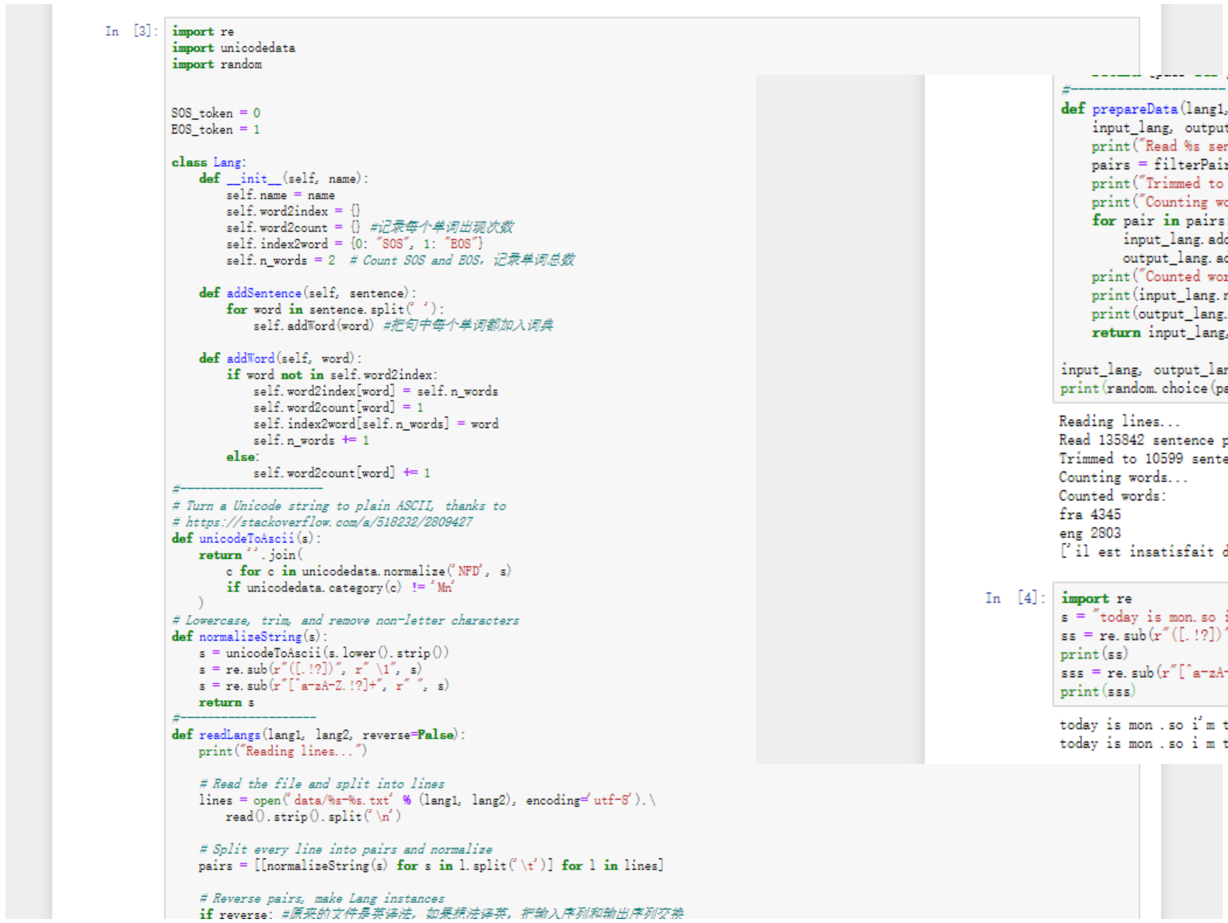
20210127汇报

孙阳

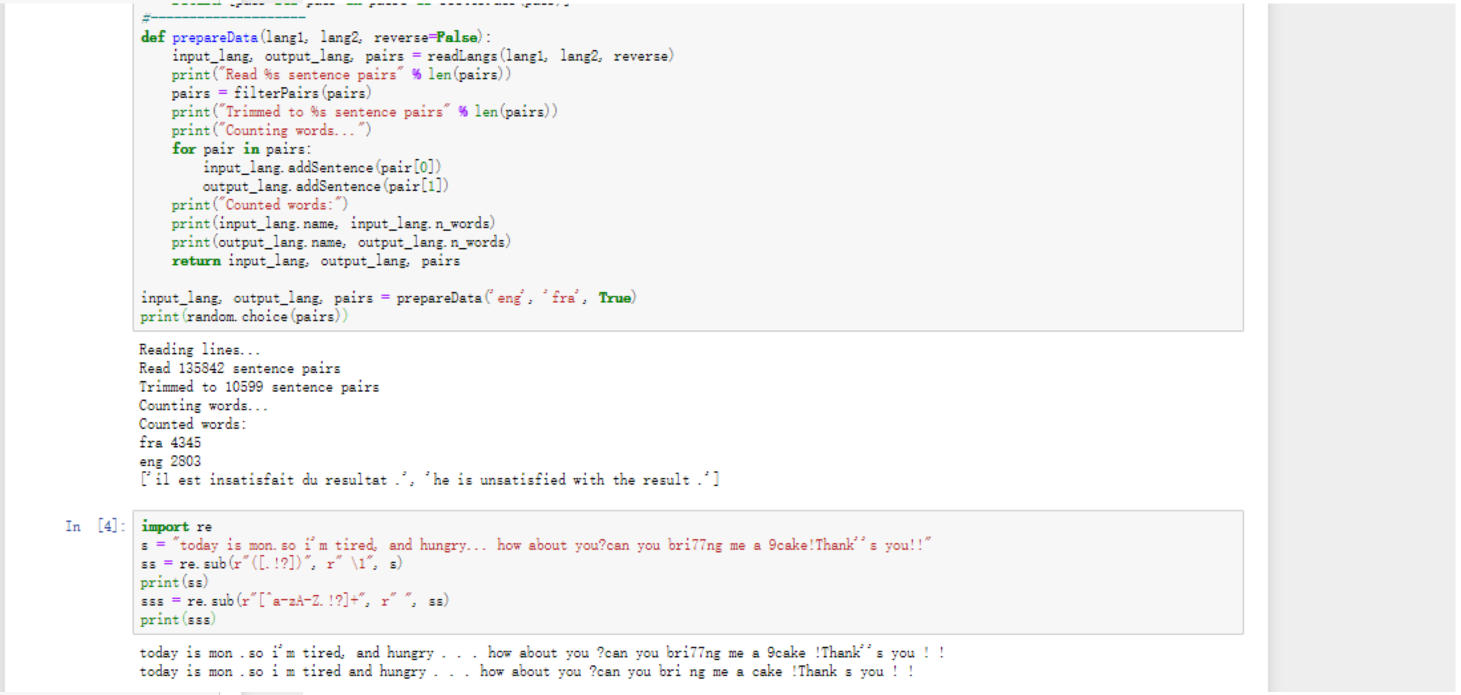
实验部分

- 李宏毅hw8 seq2seq 的前半部分

实验部分



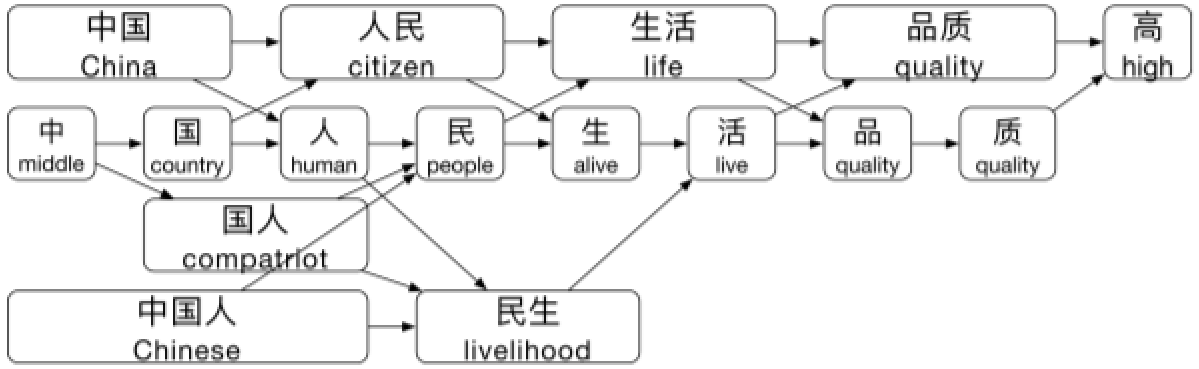
```
In [3]: import re
        import unicodedata
        import random


        SOS_token = 0
        EOS_token = 1

        class Lang:
            def __init__(self, name):
                self.name = name
                self.word2index = {}
                self.word2count = {} #记录每个单词出现次数
                self.index2word = {0: "SOS", 1: "EOS"}
                self.n_words = 2  # Count SOS and EOS, 记录单词总数

            def addSentence(self, sentence):
                for word in sentence.split(' '):
                    self.addWord(word) #把句中每个单词都加入词典

            def addWord(self, word):
                if word not in self.word2index:
                    self.word2index[word] = self.n_words
                    self.word2count[word] = 1
                    self.index2word[self.n_words] = word
                    self.n_words += 1
                else:
                    self.word2count[word] += 1
        #---------------------
        # Turn a Unicode string to plain ASCII, thanks to
        # https://stackoverflow.com/a/518232/2809427
        def unicodeToAscii(s):
            return ''.join(
                c for c in unicodedata.normalize('NFD', s)
                if unicodedata.category(c) != 'Mn'
            )
        # Lowercase, trim, and remove non-letter characters
        def normalizeString(s):
            s = unicodeToAscii(s.lower().strip())
            s = re.sub(r"([.!?])", r" \1", s)
            s = re.sub(r"[^a-zA-Z.!?]+", r" ", s)
            return s
        #--------------------
        def readLangs(lang1, lang2, reverse=False):
            print("Reading lines...")

            # Read the file and split into lines
            lines = open('data/%s-%s.txt' % (lang1, lang2), encoding='utf-8').\
                read().strip().split('\n')

            # Split every line into pairs and normalize
            pairs = [[normalizeString(s) for s in l.split('\t')] for l in lines]

            # Reverse pairs, make Lang instances
            if reverse: #原来的文件是英译法，如果想法译英，把输入序列和输出序列交换
```



```
        #--------------------
        def prepareData(lang1, lang2, reverse=False):
            input_lang, output_lang, pairs = readLangs(lang1, lang2, reverse)
            print("Read %s sentence pairs" % len(pairs))
            pairs = filterPairs(pairs)
            print("Trimmed to %s sentence pairs" % len(pairs))
            print("Counting words...")
            for pair in pairs:
                input_lang.addSentence(pair[0])
                output_lang.addSentence(pair[1])
            print("Counted words:")
            print(input_lang.name, input_lang.n_words)
            print(output_lang.name, output_lang.n_words)
            return input_lang, output_lang, pairs

        input_lang, output_lang, pairs = prepareData('eng', 'fra', True)
        print(random.choice(pairs))

        Reading lines...
        Read 135842 sentence pairs
        Trimmed to 10599 sentence pairs
        Counting words...
        Counted words:
        fra 4345
        eng 2803
        ['il est insatisfait du resultat .', 'he is unsatisfied with the result .']

In [4]: import re
        s = "today is mon.so i'm tired, and hungry... how about you?can you bri77ng me a 9cake!Thank''s you!!"
        ss = re.sub(r"([.!?])", r" \1", s)
        print(ss)
        sss = re.sub(r"[^a-zA-Z.!?]+", r" ", ss)
        print(sss)

        today is mon .so i'm tired, and hungry . . . how about you ?can you bri77ng me a 9cake !Thank''s you ! !
        today is mon .so i m tired and hungry . . . how about you ?can you bri ng me a cake !Thank s you ! !
```

论文学习

- 《Lattice CNNs for Matching Based Chinese Question Answering》用于中文问答匹配的一种基于lattice的CNN模型——2019AAAI
- 解决由于中文分词不当产生的歧义问题。



lattice CNNs

- 句子表示可以是原始CNN，也可以是Lattice CNN。在原始CN中，卷积核按照顺序扫面每n-gram，并得到一个特征向量，该向量可以看作是中心词的表示，被传递至下一层。但是，每一个词在每一个lattice中可能具有不同粒度的上下文词，并且可以被视为具有相同长度的卷积核的中心。因此，不同于原始CNN，lattice CNN对于一个词可能产生多个特征向量。
- “citizen”具有四个长度为3的上下文的中心词（China -citizen - life, China - citizen - alive, country - citizen -life, country - citizen - alive）

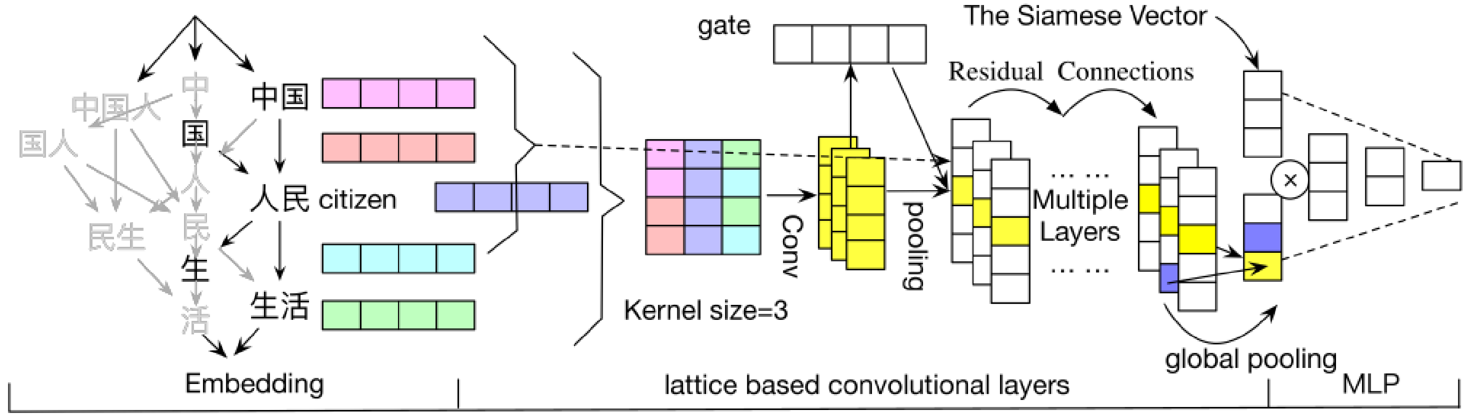
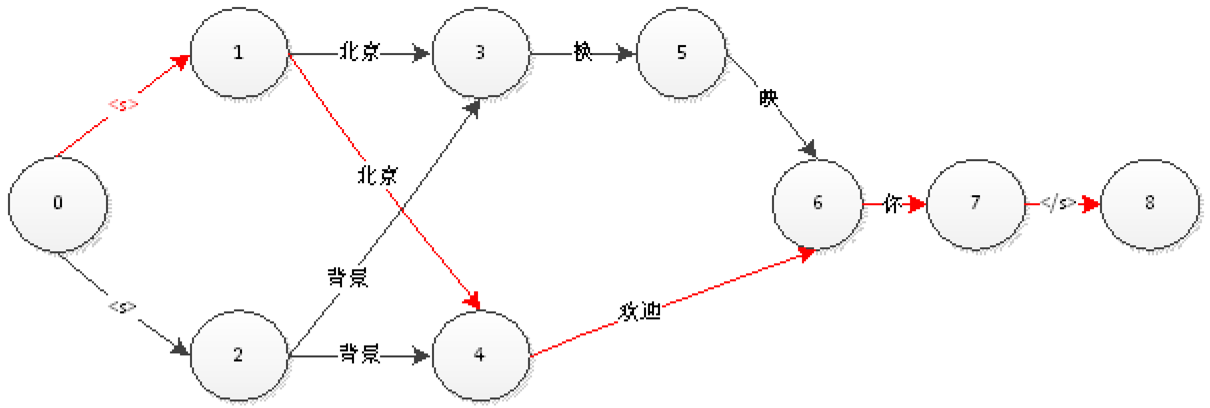


Figure 2: An illustration of our LCN-gated, when “people” is being considered as the center of convolutional spans.

Word lattice

- 。

词网格本质是一个有向无环图G=<V，每个词网格包含一个仅有入度为0的开始结点<s>以及一个出度为0结束结点</s>



lattice based CNN layer

- 在论文中，作者并不直接采用标准CNN接受word lattice作为输入，而是提出了一种基于lattice的CNN层，允许标准CNN处理Word lattice输入。
- 特征向量表达式：

$$F_w = g\{f(\boldsymbol{W}_c(\boldsymbol{v}_{\boldsymbol{w}_1} : \ldots : \boldsymbol{v}_{\boldsymbol{w}_n}) + \boldsymbol{b}_c^T) | \forall i, w_i \in V, (w_i, w_{i+1}) \in E, w_{\lceil \frac{n+1}{2} \rceil} = w\} \tag{3}$$

- 门池化操作：

$$\alpha_1, \ldots, \alpha_t = \texttt{softmax}\{\boldsymbol{v}_g^T \boldsymbol{v}_1 + b_g, \ldots, \boldsymbol{v}_g^T \boldsymbol{v}_t + b_g\} \tag{4}$$

$$\texttt{gated-pooling}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t\} = \sum_{i=1}^{n} \alpha_i \times \boldsymbol{v}_i \tag{5}$$