神经语言模型

题目：利用给定语料库（或者自选语料库），利用神经语言模型（如：Word2Vec， GloVe 等模型）来训练词向量，通过对词向量的聚类或者其他方法来验证词向量的有效性。 截至日期：5 月 20 日晚 12 点前。

1 词向量

词向量技术是将词转化成为稠密向量，并且对于相似的词，其对应的词向量也相近。在自然语言处理任务中，首先需要考虑词如何在计算机中表示。通常，有两种表示方式：one-hot representation 和 distribution representation。

1.1 离散表示

传统的基于规则或基于统计的自然语义处理方法将单词看作一个原子符号被称作 one-hot representation。one-hot representation 把每个词表示为一个长向量。这个向量的维度是词表大小，向量中只有一个维度的值为 1，其余维度为 0，这个维度就代表了当前的词。例如：

苹果 [0，0，0，1，0，0，0，0，0，……]。one-hot representation 相当于给每个词分配一个 id，这就导致这种表示方式不能展示词与词之间的关系。另外，one-hot representation 将会导致特征空间非常大，但也带来一个好处，就是在高维空间中，很多应用任务线性可分。

1.2 分布式表示

word embedding 指的是将词转化成一种分布式表示，又称词向量。分布式表示将词表示成一个定长的连续的稠密向量。

1. 分布式表示优点：

词之间存在相似关系：是词之间存在“距离”概念，这对很多自然语言处理的任务非常有帮助。

2. 包含更多信息：

词向量能够包含更多信息，并且每一维都有特定的含义。在采用 one-hot 特征时，可以对特征向量进行删减，词向量则不能。

2 生成词向量的生成

生成词向量的方法有很多，这些方法都依照一个思想：任一词的含义可以用它的周边词来表示。生成词向量的方式可分为：基于统计的方法和基于语言模型(language model)的方法。

2.1 基于统计方法

共现矩阵：通过统计一个事先指定大小的窗口内的 word 共现次数，以 word 周边的共现词的次数做为当前 word 的 vector。具体来说，我们通过从大量的语料文本中构建一个共现矩阵来定义 word representation。

例如，有语料如下：

I like deep learning.

I like NLP.

I enjoy flying.

则其共现矩阵如下：

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

矩阵定义的词向量在一定程度上缓解了 one-hot 向量相似度为 0 的问题，但没有解决数据稀疏性和维度灾难的问题。

SVD 奇异值分解

既然基于 co-occurrence 矩阵得到的离散词向量存在着高维和稀疏性的问题，一个自然而然的解决思路是对原始词向量进行降维，从而得到一个稠密的连续词向量。进行 SVD 分解，得到矩阵正交矩阵 U，对 U 进行归一化得到矩阵如下：

I	0.24	0.21	0.10	0.38	-0.18	-0.18	-0.42	-0.06
like	0.20	0.82	-0.17	0.31	0.18	-0.23	0.13	0.14
enjoy	0.37	0.64	0.16	0.00	-0.58	0.64	0.00	-0.31
deep	0.36	0.38	0.35	-0.07	0.45	0.08	0.55	-0.47
learning	0.40	0.52	-0.50	-0.43	0.35	0.16	-0.47	-0.40
NLP	0.35	0.35	-0.22	-0.19	0.13	0.49	0.21	0.66
flying	0.41	0.42	-0.40	-0.38	-0.51	-0.43	0.42	-0.12
.	0.38	0.58	0.59	-0.62	-0.03	-0.23	-0.26	0.24

SVD 得到了 word 的稠密（dense）矩阵，该矩阵具有很多良好的性质：语义相近的词在向量空间相近，甚至可以一定程度反映 word 间的线性关系。

3 Word2Vec 模型

Word2Vec 是语言模型中的一种，它是从大量文本预料中以无监督方式学习语义知识的模型，被广泛地应用于自然语言处理中。Word2Vec 是用来生成词向量的工具，而词向量与语言模型有着密切的关系。

3.1 统计语言模型

统计语言模型是用来计算一个句子的概率的概率模型，它通常基于一个语料库来构建。假设 $W=(w_1,w_2,\ldots,w_T)$ 表示由 T 个词 $w_1,w_2,\ldots,w_T$ 按顺序构成的一个句子，则 $w_1,w_2,\ldots,w_T$ 的联合概率为：

$$p(W)=p(w_1,w_2,\ldots,w_T) \tag{3.1}$$

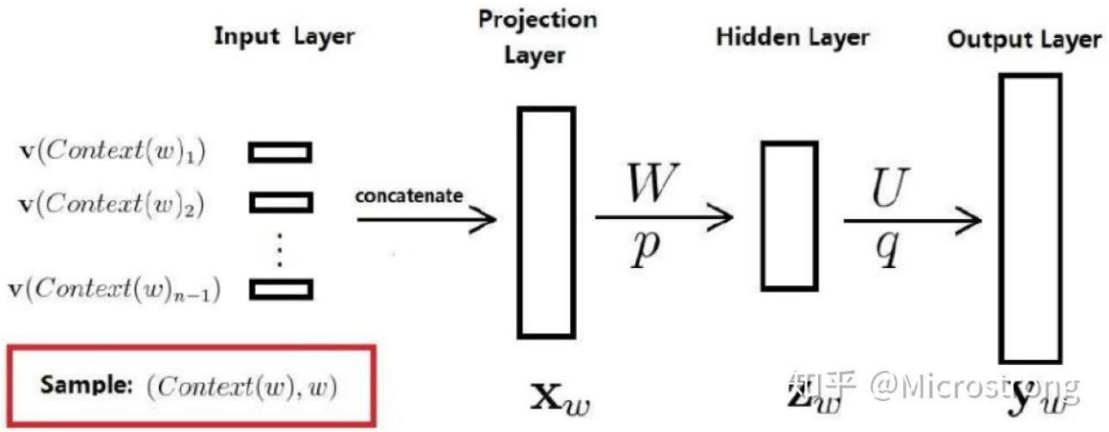
p(W)被称为语言模型，即用来计算这个句子概率的模型。利用 Bayes 公式，上式可以被链式地分解为:

$$p(W)=p(w_1)p(w_2|w_1)p(w_3|w_1,w_2)\ldots p(w_T|w_1,w_2,\ldots,w_{T-1})$$

其中的条件概率就是语言模型的参数，若这些参数已经全部算得，那么给定一个句子 W，就可以很快地计算出相应地概率 p(W)了。

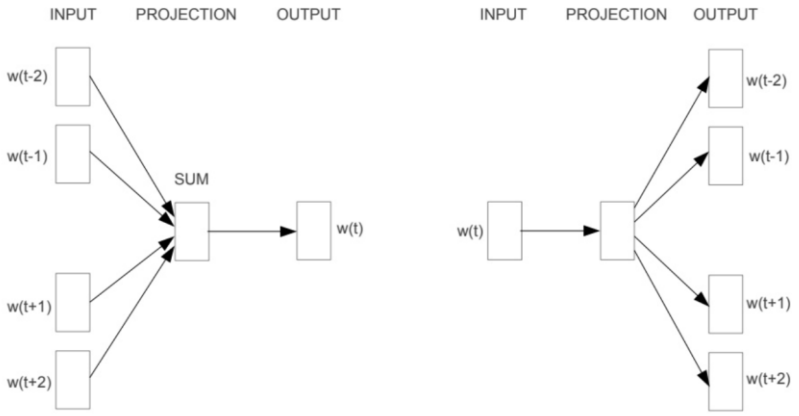
3.2 神经概率语言模型

下图给出了神经网络的结构示意图。模型一共三层，第一层是映射层，将 n 个单词映射为对应 word embeddings 的拼接，其实这一层就是 MLP 的输入层；第二层是隐藏层，激活函数用 tanh；第三层是输出层，因为是语言模型，需要根据前 n 个单词预测下一个单词，所以是一个多分类器，用 Softmax。整个模型最大的计算量集中在最后一层上，因为一般来说词汇表都很大，需要计算每个单词的条件概率，是整个模型的计算瓶颈。



3.3 Word2Vec 的网络结构

Word2Vec 是轻量级的神经网络，其模型仅仅包括输入层、隐藏层和输出层，模型框架根据输入输出的不同，主要包括 CBOW 和 Skip-gram 模型。 CBOW 的方式是在知道词 w_t 的上下文 $w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$ 的情况下预测当前词 w_t .而 Skip-gram 是在知道了词 w_t 的情况下,对词 w_t 的上下 文 $w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$ 进行预测，如下图所示：



3.4 Word2Vec 在工业界的应用

Word2Vec 主要原理是根据上下文来预测单词，一个词的意义往往可以从其前后的句子中抽取出来。而用户的行为也是一种相似的时间序列，可以通过上下文进行推断。当用户浏览并与内容进行交互时，我们可以从用户前后的交互过程中判断行为的抽象特征，这就使得我们可以把词向量模型应用到推荐、广告领域当中。

NLP 领域：

1) Word2Vec 学习到的词向量代表了词的语义，可以用来做分类、聚类、也可以做词的相似度计算。
2) 把 Word2Vec 生成的向量直接作为深度神经网络的输入，可以做 sentiment analysis 等工作。

4 实验过程

4.1 实验预处理

与前一次实验相同，删去所有的隐藏符号，删除所有的非中文字符，不考虑上下文关系的前提下删去所有标点符号。以 jieba 库对中文语料进行分词。由实验要求得，需对数据库进行训练集的选取。在该实验当中，将测试集化为 2000 行语料，其余行列为训练集。

```
def get_data():
    """如果文档还没分词，就进行分词"""
    outfilename_1 = "./cnews.train_jieba.txt"
    outfilename_2 = "./cnews.test_jieba.txt"
    if not os.path.exists('./cnews.train_jieba.txt'):
        outputs = open(outfilename_1, 'w', encoding='UTF-8')
        outputs_test = open(outfilename_2, 'w', encoding='UTF-8')
        datasets_root = "./datasets_"
        catalog = "inf.txt"

        test_num = 10
        test_length = 20
        with open(os.path.join(datasets_root, catalog), "r", encoding='utf-8') as f:
            all_files = f.readline().split(",")
            print(all_files)

        for name in all_files:
            with open(os.path.join(datasets_root, name + ".txt"), "r", encoding='utf-8') as f:
                file_read = f.readlines()
                train_num = len(file_read) - test_num
                choice_index = np.random.choice(len(file_read), test_num + train_num, replace=False)
```

4.2 运用 Word2Vec 进行训练

本实验选用 gensim.models 包中的 word2vec 模块。

```
fr = open('./cnews.train_jieba.txt', 'r', encoding='utf-8')
train = []
for line in fr.readlines():
    line = [word.strip() for word in line.split(' ')]
    train.append(line)

num_features = 300  # Word vector dimensionality
min_word_count = 10  # Minimum word count
num_workers = 16  # Number of threads to run in parallel
context = 10  # Context window size
downsampling = 1e-3  # Downsample setting for frequent words
sentences = models.word2vec.Text8Corpus("cnews.train_jieba.txt")

model = models.word2vec.Word2Vec(sentences, workers=num_workers,
                         vector_size=num_features, min_count=min_word_count,
                         window=context, sg=1, sample=downsampling)
model.init_sims(replace=True)
# 保存模型，供日後使用
model.save("model201708.model")
```

4.3 实验结果

这里选取在各个小说中出现的人物名称以及武术派别等名词，使用训练好的模型输出与其词向量最接近的十个词，通过观察它们之间的联系，检查语言模型训练的好坏。

```
model=models.word2vec.Word2Vec.load("model201708.model")
print(model.wv.most_similar("乔峰", topn=10))
print(model.wv.most_similar("阿紫", topn=10))
print(model.wv.most_similar("杨过", topn=10))
print(model.wv.most_similar("郭靖", topn=10))
print(model.wv.most_similar("虚竹", topn=10))
#print(model.wv["乔峰"])
#os.system("pause")
```

实验结果如下：

```
[('马夫人', 0.6235629916191101), ('徐长老', 0.6182544231414795), ('白世镜', 0.5952910780906677)
[('游坦之', 0.6500557065010071), ('紫', 0.5712727904319763), ('阿紫道', 0.5371442437171936), (
[('小龙女', 0.7259780168533325), ('金轮法王', 0.6275424361228943), ('过儿', 0.5829077959060669)
[('黄蓉', 0.6137715578079224), ('柯镇恶', 0.5291078686714172), ('拖雷', 0.5288021564483643), (
```

名词	相关词
郭靖	黄蓉、托雷、穆念慈、欧阳锋
杨过	小龙女、金轮法王、过儿、神雕侠侣、公孙谷主
灭绝师太	峨嵋派、丁敏君、静玄、纪晓芙、宋青书
倚天剑	屠龙刀、屠龙、此剑、削断、争锋
屠龙刀	屠龙、倚天剑、宝刀、至尊、金毛狮、谢大侠
西施	范蠡、范大夫、杨贵妃、夫差、美女

从实验结果可以看出，相关词基本具有一定的关联性，说明语言模型训练正确。