

成 绩

****

深度学习与自然语言处理

第四次大作业

|  |  |
| --- | --- |
| 院（系）名称 | 自动化科学与电气工程学院 |
| 专业名称 | 电子信息 |
| 学号 | ZY2103113 |
| 姓名 | 孙茗逸 |
| 指导教师 | 秦曾昌 |

2022年5月19日

1. **任务描述**

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。

1. **实验原理**

计算机无法看懂人类的自然语言，也无法对其进行直接处理，因此自然语言处理第一步需要将自然文字转换为计算机能够“看懂”的数字，因此需要将自然文字进行编码，转换由数字组成的词向量。

1. **词向量的表示形式**
2. One-hot编码

One-hot编码是一种简单的词向量表示形式。具体地，用很长的一个向量来表示一个词，其中向量的长度为词典的大小；向量的所有分量只有一个为“1”，其余全为“0”，为“1”的位置为该词在词典中的字典序。例如，当总词典中只有“我”“爱”“北航”三个词时，三个词语可以分别编码为“100”、“010”、“001”。

One-hot编码简单易表示。然而，词典总次数很大时，one-hot向量会很长，容易造成维度灾难；此外，one-hot编码将每个词看成独立的个体，忽略了词与词之间的联系（如语义相近的词、语义相反的词等）。

如果将自然语言的每一个词映射成一个固定长度的短向量，将所有这些向量放在一起形成一个词向量空间，而每一向量则为该空间中的一个点，在这个空间上引入“距离”，则可以根据词之间的距离来判断它们之间的语义相似性。这便是词语的分布式表示（distributed representation）。

1. 分布式表示（Distributed Representation）

上述所说，将不同的自然词语映射为词向量空间中的不同向量的做法，便是词的分布式表示。分布式表示引入了“距离”的概念从而可以衡量词与词之间的联系，这对建模自然语言的语义信息大有裨益；此外，有一个多维的词向量而非只包含0、1的one-hot向量对词语建模，使得向量可以包含更为丰富的语义信息。

Word2Vec采用的是分布式表示的词向量。

1. **Word2Vec**

作为轻量级神经网络，Word2Vec模型包括输入层、隐藏层和输出层，主要分为CBOW和Skip-gram两种模型。其中，CBOW模型在已知上下文的情况下预测当前词，其中滑动窗口的大小为；Skip-gram模型在已知当前词的情况下对上下文进行预测。CBOW模型和Skip-gram模型分别如图1和图2所示。

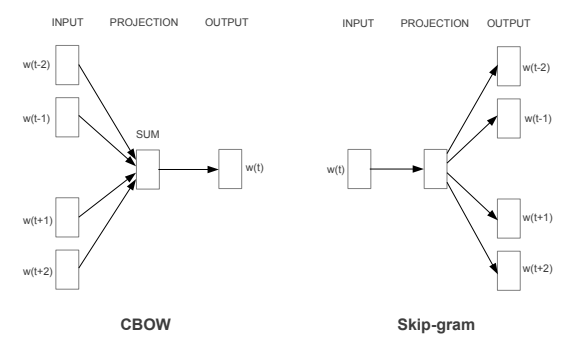


图1 CBOW模型示意图 图2 Skip-gram模型示意图

1. **K-Means聚类**

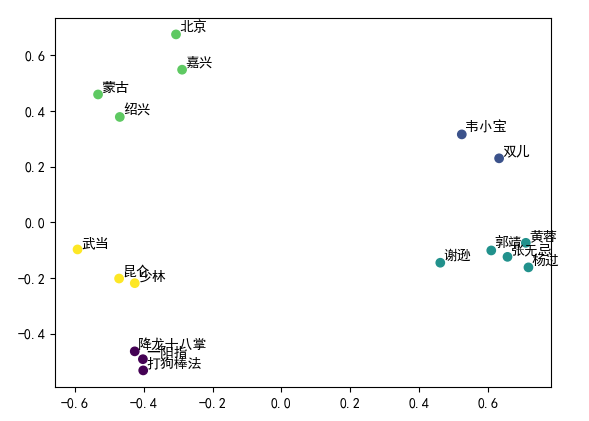
聚类是将物理或抽象对象的集合分成由类似的对象组成的多个类的过程。K-Means 是一种典型的聚类算法，算法步骤如下：

1. 选取K个点做为初始聚集的簇心（也可选择非样本点）;
2. 分别计算每个样本点到 K个簇核心的距离（这里的距离一般取欧氏距离或余弦距离），找到离该点最近的簇核心，将它归属到对应的簇；
3. 所有点都归属到簇之后，M个点就分为了 K个簇。之后重新计算每个簇的重心（平均距离中心），将其定为新的“簇核心”；
4. 反复迭代 2 - 3 步骤，直到达到中止条件（迭代次数、最小平方误差、簇中心点变化率等）。
5. **实验内容与结果**
6. **实验内容**

​ 本文基于 Python 进行 Word2vec 模型的训练。首先使用jieba对16本金庸小说的合集进行分词，分词处理与前几次作业基本相同，在此不再赘述。分词分句完成后，用gensim中的word2vec进行词嵌入，并打包输出键为单词，值为词向量的字典。得到单词对应的词向量后，对其标准化，再使用 K-Means 对指定单词的词向量进行聚类分析，并将词向量使用 PCA 降维后绘图。

1. **实验结果**

​ 输入keys=['郭靖','黄蓉','杨过','张无忌','谢逊','韦小宝','双儿','少林','武当','昆仑','北京','蒙古','嘉兴','绍兴','降龙十八掌','打狗棒法','一阳指']，选择 K-Means 簇数为 5，得词向量聚类结果如下图：



**附录：代码**

import jieba

import re

from gensim.models import word2vec

import numpy as np

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

import pickle as pkl

def cut\_sentences(content):

"""文本分句处理"""

end\_flag = ['?', '!', '.', '？', '！', '。', '…']

content\_len = len(content)

sentences = []

tmp\_char = ''

for idx, char in enumerate(content):

# 拼接字符

tmp\_char += char

# 判断是否已经到了最后一位

if (idx + 1) == content\_len:

sentences.append(tmp\_char)

break

# 判断此字符是否为结束符号

if char in end\_flag:

# 再判断下一个字符是否为结束符号，如果不是结束符号，则切分句子

next\_idx = idx + 1

if not content[next\_idx] in end\_flag:

sentences.append(tmp\_char)

tmp\_char = ''

return sentences

def word\_seg(path, dest):

"""对path指向文本进行基于jieba的分词，返回以空格分隔单词的文本"""

with open(path, "r", encoding="ANSI") as f:

data = f.read()

f.close()

text = cut\_sentences(data)

with open(dest, "w+", encoding="utf-8") as f:

for sentence in text:

sentence = data = re.sub('[^\u4e00-\u9fa5]+', '', sentence)

f.write(" ".join(jieba.lcut(sentence, use\_paddle=True, cut\_all=False)) + '\n')

def vec\_gen(path):

"""根据分句分词完成的文件路径生成词向量字典"""

train\_data = word2vec.LineSentence(path)

model = word2vec.Word2Vec(train\_data,

vector\_size=100,

window=5,

workers=4)

model.wv.vectors = model.wv.vectors / (np.linalg.norm(model.wv.vectors, axis=1).reshape(-1, 1))

vec\_dist = dict(zip(model.wv.index\_to\_key,model.wv.vectors))

with open('VEC/vec\_dist', 'wb') as f:

pkl.dump(vec\_dist,f)

def cluster(keys, n\_clusters):

with open('VEC/vec\_dist', 'rb') as f:

vec\_dist = pkl.load(f)

vec = []

for k in keys:

vec.append(vec\_dist[k])

label = KMeans(n\_clusters=n\_clusters).fit\_predict(vec)

vec = PCA(n\_components=2).fit\_transform(vec)

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

plt.scatter(vec[:,0],vec[:, 1],c=label)

for i, w in enumerate(keys):

plt.annotate(s=w, xy=(vec[:, 0][i], vec[:, 1][i]),

xytext=(vec[:, 0][i] + 0.01, vec[:, 1][i] + 0.01))

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

# s = word\_seg('VEC/text/merged.txt','VEC/seg.txt')

# wv = vec\_gen('VEC/seg.txt')

cluster(['郭靖','黄蓉','杨过','小龙女','郭襄','张无忌','谢逊','韦小宝','双儿','康熙','少林','武当','昆仑','北京','嘉兴','杭州','扬州','降龙十八掌','打狗棒法','一阳指'], 5)