词向量模型

一 实验内容

利用给定语料库（或者自选语料库），利用神经语言模型 Word2Vec 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。

二 背景知识

2.1 word embedding

词向量（Word embedding），又叫 Word 嵌入式，是自然语言处理（NLP）中的一组语言建模和特征学习技术的统称，其中来自词汇表的单词或短语被映射到实数的向量。从概念上讲，它涉及从每个单词一维的空间到具有更低维度的连续向量空间的数学嵌入。

如果将 word 看作文本的最小单元，可以将 Word Embedding 理解为一种映射，其过程是：将文本空间中的某个 word，通过一定的方法，映射或者说嵌入（embedding）到另一个数值向量空间（之所以称之为 embedding，是因为这种表示方法往往伴随着一种降维的意思）。

生成这种映射的方法包括神经网络，单词共生矩阵的降维，概率模型，可解释的知识库方法，和术语的显式表示单词出现的背景。

当用作底层输入表示时，单词和短语嵌入已经被证明可以提高 NLP 任务的性能，例如语法分析和情感分析。

Word Embedding 可分为：（1）基于频率的 Word Embedding（Frequency based embedding）主要有：Count Vector，TF-IDF Vector 和 Co-Occurrence Vector

（2）基于预测的 Word Embedding（Prediction based embedding）。主要包括：CBOW（continues bag of words）以及 Skip–Gram。

2.2 Word2Vec

Word2Vec 是 Google 公司于 2013 年发布的一个开源词向量工具包。该项目的算法理论参考了 Bengio 在 2003 年设计的神经网络语言模型。由于此神经网络模型使用了两次非线性变换，网络参数很多，训练缓慢，因此不适合大语料。Mikolov 团队对其做了简化，实现了 Word2Vec 词向量模型。它简单、高效，特

别适合从大规模、超大规模的语料中获取高精度的词向量表示。因此，项目一经发布就引起了业界的广泛重视，并在多种 NLP 任务中获得了良好的效果，成为 NLP 在语义相似度计算中的重大突破。Word2Vec 及同类的词向量模型都基于如下假设：衡量两个词在语义上的相似性，决定于其邻居词分布是否类似。显然这是源于认知语言学中的“距离象似性”原理：词汇与其上下文构成了一个“象”。当从语料中训练出相同或相近的两个“象”时，无论这两个“象”的中心词汇在字面上是否一致，它们在语义上都是相似的。

2.3 skip-grammodel

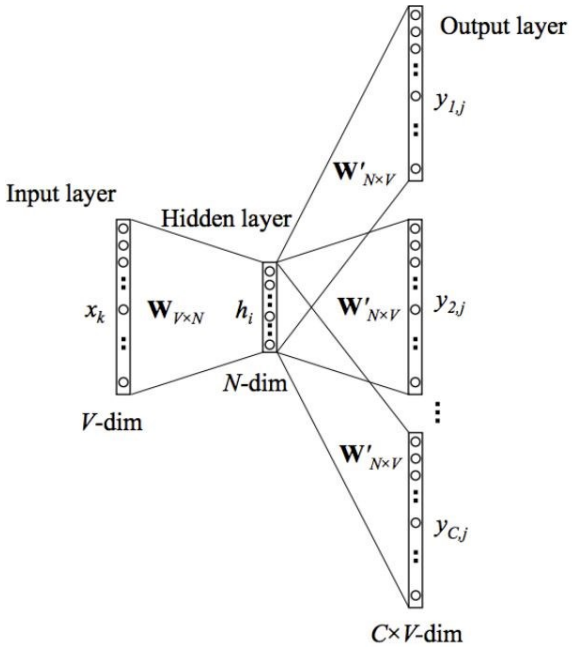

图 1skip-grammodel

在上图中，输入向量 x 代表某个单词的 one-hot 编码，对应的输出向量 $\{y_1,\ldots,y_C\}$。输入层与隐藏层之间的权重矩阵 W 的第 i 行代表词汇表中第 i 个单词的权重。这个权重矩阵 W 就是我们需要学习的目标(同 W')，因为这个权重矩阵包含了词汇表中所有单词的权重信息。上述模型中，每个输出单词向量Q也有个 $N\times V$ 维的输出向量 W。最后模型还有 N 个结点的隐藏层，我们可以发现隐藏层节点 h_i 的输入就是输入层输入的加权求和。因此由于输入向量 x 是one-hot编码，那么只有向量中的非零元素才能对隐藏层产生输入。因此对于输入向量 x 其中 $x_k=1$，所以隐藏层的输出只与权重矩阵第 k 行相关。

2.4 CBOW model

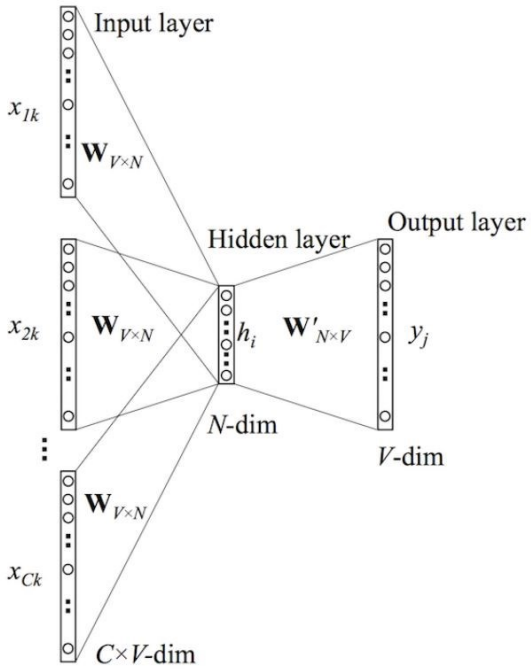

图 2CBOWmodel

在上图中，该模型的输入输出与 skip-gram 模型的输入输出是相反的。这里输入层是由 one-hot 编码的输入上下文$\{x_1,\ldots,x_C\}$组成，其中窗口大小为C，词汇表大小为V。隐藏层是N维的向量。最后输出层是也被 one-hot 编码的输出单词y。被 one-hot 编码的输入向量通过一个$V\times N$维的权重矩阵W连接到隐藏层;隐藏层通过一个$N\times V$的权重矩阵W'连接到输出层。

三 实验流程

3.1 文本预处理

主要包括语料库的读取、停词和标点符号的过滤以及分词三部分。其中前两部分采用前几次作业的源码进行操作。第三部分由于金庸小说中有很多独特的人名门派，比如“郭靖”,“黄药师”等等，需要扩充分词库，从而实现准确的分词。这里借鉴了[1]的方法，从金庸网-金庸数据库中下载了武功、门派、人名的 txt 文件，用于分词处理。

3.2 训练词向量

采用 gensim 模块训练词向量，它提供了 Word2Vec 模型以及接口。采用

CBOW 和 skip-gram 模型分别进行训练，并对两种模型的训练结果进行比较。

3.3 词向量聚类和比较

Word2Vec 模型训练好后，得到的词向量可以进行聚类比较分析。Gensim 模块也提供了相关接口。

除了 gensim 模块提供的比较分析方法，还借鉴了[1]中所使用的对应比较函数，即 x1 相对于 y1 的关系和 x2 相对于 y2 的关系是相近的，已知 x1、y1 和 x2，求 y2。

四 实验结果

分别使用 CBOW 模型和 skip-gram 模型，向量长度为 300，迭代次数为 10 次，进行人物、武功、地名词汇的比较。

4.1 人物关系的比较

采用了我印象相对深刻的 6 位人物，包括 3 名男性角色和 3 名女性角色，取相关性排名前 5 的人物。

表 1 CBOW 模型结果

	1	2	3
段誉	胡斐	赵敏	石破天
郭靖	黄蓉	杨过	胡斐
张无忌	令狐冲	石破天	胡斐
王语嫣	赵敏	郭芙	盈盈
黄蓉	郭靖	杨过	胡斐
赵敏	盈盈	林平之	王语嫣

表 2 skip-gram 模型结果

	1	2	3
段誉	王语嫣	慕容复	木婉清
郭靖	黄蓉	欧阳克	杨康
张无忌	周芷若	赵敏	张翠山

王语嫣	段誉	阿碧	钟灵
黄蓉	洪七公	郭靖	蓉儿
赵敏	周芷若	小昭	蛛儿

比较 CBOW 和 skip-gram 的结果可以发现，skip-gram 更倾向于建立本书中的人物关联，一同出现的频率越高相关性越大，而 CBOW 可以联系不同书中的人物，去寻找性格相近的人物进行匹配。

4.2 武功关系的比较

采用了北冥神功、化功大法、吸星大法三种同源的武功，观察它们相关性最大的词汇。

	1	2	3
北冥神功	一阳指	空明拳	乾坤大挪移
化功大法	一阳指	吸星大法	玄冥神掌
吸星大法	易筋经	六脉神剑	七伤拳

与化功大法相似度比较高的包括了吸星大法和玄冥神掌两门阴毒的武功，说明模型对正派武功和旁门左道是有一定区分的。吸星大法最相近的是易筋经，是因为只有易筋经才能化解吸星大法带来的损害。

4.3 武功和人物关系

降龙十八掌之于郭靖相当于黄蓉的哪门武功招式呢？模型给出的答案是打狗棒法，这符合小说的情节设定。

4.4 地名间的关系

给定“桃花岛”、“大理”、“西夏”三个小说中常出现的地名，寻找相近的地名设定。

给定项	匹配项
桃花岛	姑苏
襄阳	山海关
西夏	云南

风景秀美的桃花岛让人也不禁联想到茶花盛开的姑苏。郭靖夫妇把守的襄阳一夫当关万夫莫开，与山海关的地理要塞也有相近之处。西夏和云南分别处于西

北和西南，小说中都是少数民族统治下的邦国。

五 参考链接

1

https://blog.csdn.net/weixin_50891266/article/details/116750204?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1-116750204-blog-124732760.pc_relevant_aa&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1-116750204-blog-124732760.pc_relevant_aa&utm_relevant_index=2

五 相关代码

```
import os
import jieba
import warnings
import gensim.models as w2v
import re
import pdb


class TraversalFun():

    # 1 初始化
        def __init__(self, rootDir, stop_words):
            self.rootDir = rootDir
            self.stop_words = stop_words

    def TraversalDir(self):
        return self.getCorpus()

    def getCorpus(self):
        rootDir = self.rootDir
        stop_words = self.stop_words
        corpus = []
                r1 = u'[a-zA-Z0-9’!"#$%&\'()*+,-./:：;<=>?@，?★、…【】《》？“”‘’！[\\]^_`{|}~]+' # 用户也可以在此进行自定义过滤字符
        listdir = os.listdir(rootDir)
        count=0
        for file in listdir:
            path = os.path.join(rootDir, file)
            if os.path.isfile(path):
                with open(os.path.abspath(path), "r", encoding='gbk', errors="ignore") as file:
```

```
                filecontext = file.read()
                filecontext = re.sub(r1, '', filecontext)
                filecontext = filecontext.replace("\n", '')
                filecontext = filecontext.replace(" ", '')
                filecontext = filecontext.replace("\u3000", '')
                filecontext = filecontext.replace("本书来自www.cr173.com免费txt小说下载站\n更多更新免费电子书请关注www.cr173.com", '')
                filecontext = filecontext.replace("本书来自免费小说下载站\n更多更新免费电子书请关注", '')
                # for word in stop_words:
                #     filecontext = filecontext.replace(word, '')
                filecontext = filecontext.split('。')
                #seg_list = jieba.cut(filecontext, cut_all=True)
                #corpus += seg_list
                count += len(filecontext)
                corpus.extend(filecontext)
        return corpus,count

def get_words(filename):
        with open(filename, "r", encoding='gbk', errors="ignore") as f:
        words = [word.strip() for word in f.readlines()]
    return words

def add_words(words):
    for word in words:
        jieba.add_word(word)
    return

def train_model(sentences):
        model = w2v.Word2Vec(sentences=sentences, min_count=5, size=300, window=5, sg=1, iter=10)
    model.save('./CBOW.model') # 保存模型
    model.wv.save_word2vec_format("./CBOW_300vec.txt", binary=False)

def find_relation(model, a, b, c):
        d, _ = model.wv.most_similar(positive=[c, b], negative=[a])[0]
    print (c,d)
```

```
def train():
    stop_words = get_words("./人物武功门派和停词/stop_words.txt"); stop_words.remove('。')
    menpai = get_words("./人物武功门派和停词/金庸小说全门派.txt")
    renwu = get_words("./人物武功门派和停词/金庸小说全人物.txt")
    wugong = get_words("./人物武功门派和停词/金庸小说全武功.txt")
                    add_words(menpai); add_words(renwu); add_words(wugong)
        tra = TraversalFun("./datasets", stop_words)
    corpus,count = tra.TraversalDir()
    sentences = [jieba.lcut(sentence) for sentence in corpus]


    model = train_model(sentences)

def test():
                                model = w2v.Word2Vec.load("./CBOW.model")
    renwu = ["杨过", "杨康", "郭靖", "段誉", "张无忌", "令狐冲", "风清扬", "林平之", "狄云", "胡斐", "小龙女", "穆念慈", "黄蓉", "王语嫣", "赵敏", "阿朱", "岳灵珊", "李莫愁", "陆无双", "程英"]
    diming = ["桃花岛", "襄阳", "昆仑", "西夏", "大理"]
    wugong = ["凌波微步", "六脉神剑", "黯然销魂掌", "降龙十八掌", "一阳指", "北冥神功", "化功大法", "吸星大法"]
    for renwu in renwu:
        print(renwu)
        print(model.wv.most_similar(renwu, topn=10))
    for item in diming:
        print(item)
        print(model.wv.most_similar(item, topn=10))
    for item in wugong:
        print(item)
        print(model.wv.most_similar(item, topn=20))

    find_relation(model, "郭靖", "降龙十八掌", "鲁有脚")


if __name__ == "__main__":
    train()
    test()
```