DL-NLP 第四次实验——Word Embedding

一、问题描述

利用 Word2Vec 模型训练 Word Embedding，根据小说中人物、武功、派别或者其他你感兴趣的特征，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、实验原理

以下实验原理来自参考文献 1、2.

1. 词向量

自然语言处理相关任务中要将自然语言交给机器学习中的算法来处理，通常需要将语言数学化，因为机器不是人，机器只认数学符号。向量是人把自然界的东西抽象出来交给机器处理的东西，基本上可以说向量是人对机器输入的主要方式了。

词向量就是用来将语言中的词进行数学化的一种方式，顾名思义，词向量就是把一个词表示成一个向量。 我们都知道词在送到神经网络训练之前需要将其编码成数值变量，常见的编码方式有两种：One-Hot Representation 和 Distributed Representation。

2. 分类器：单隐藏层的感知器

Word2vec，是一群用来产生词向量的相关模型。这些模型为浅而双层的神经网络，用来训练以重新建构语言学之词文本。网络以词表现，并且需猜测相邻位置的输入词，在 word2vec 中词袋模型假设下，词的顺序是不重要的。训练完成之后，word2vec 模型可用来映射每个词到一个向量，可用来表示词对词之间的关系，该向量为神经网络之隐藏层。

Word2Vec 主要包括 CBOW 模型（连续词袋模型）和 Skip-gram 模型（跳字模型）。

3. 停用词

停用词是指在信息检索中，为节省存储空间和提高搜索效率，在处理自然语言数据(或文本)之前或之后会自动过滤掉某些字或词,这些字或词即被称为 Stop Words（停用词)。通常意义上，停用词大致分为两类。一类是人类语言中包含的功能词，这些功能词极其普遍，与其他词相比，功能词没有什么实际含义，比如'the'、'is'、'at'、'which'、'on'等。但是对于搜索引擎来说，当所要搜索的短语包含功能词，特别是像'The Who'、'The The'或'Take The'等复合名词时，停用词的使用就会导致问题。另一类词包括词汇词，比如'want'等，这些词应用十分广泛，但是对这样的词搜索引擎无法保证能够给出真正相关的搜索结果，难以帮助缩小搜索范围，同时还会降低搜索的效率，所以通常会把这些词从问题中移去，从而提高搜索性能。

三、实验步骤与结果分析

本次实验中，主要分为：文本处理与训练数据准备、Word2vec 训练、模型测试与分析三部分。

1. 文本处理与训练数据准备

同实验一，删除压缩包中的无关文档和文件，仅留下 16 个文档文件，将每个文档中的“本书来自 www.cr173.com 免费 txt 小说下载站 更多更新免费电子书请关注 www.cr173.com”删除。本步骤为手工完成。

数据读取与预处理：

```
1.  import os
2.  import jieba
3.
4.
5.  def load_data(path, ban_stop_words=False, stop_words_path='', add_words=Fals
    e, add_words_path=''):
6.      data = []
7.      names = []
8.      stop_words = set()
9.      stop_txt = os.listdir(stop_words_path)
10.     for file in stop_txt:
11.         with open(stop_words_path + '/' + file, 'r', encoding='ANSI') as f:
```

```
            for j in f.readlines():
                stop_words.add(j.strip('\n'))
    replace = '[a-zA-Z0-9’!"#$%&\'()（）：： ‘“？、》。《，*+,-./:: ;「<=>?@，。?★、…【】《》？“”‘’！[\\]^_`{|}~]+\n\u3000 '

    add_txt = os.listdir(add_words_path)
    if add_words:
        for file in add_txt:
            with open(add_words_path + '/' + file, 'r', encoding='ANSI') as f:
                for j in f.readlines():
                    jieba.add_word(j.strip('\n'))

    files = os.listdir(path)
    for file in files:
        with open(path + '/' + file, 'r', encoding='ANSI') as f:
            t = f.read()
            for i in replace:
                t = t.replace(i, '')

            c = jieba.lcut(t)
            if ban_stop_words:
                for i in range(len(c)-1, -1, -1):
                    if c[i] in stop_words:
                        del c[i]
            data.append(c)
        f.close()
        print("{} loaded".format(file))
        names.append(file.split(".txt")[0])
    return data, names


if __name__ == '__main__':
    ban_stop_words = True
    add_words = True
    data, text_names = load_data("./data", ban_stop_words, "./stop", add_words, './words')
    text_num = len(data)

    with open('./sentences.txt', 'w', encoding='utf-8') as f:
        for i in range(len(text_names)):
            for j in data[i]:
                f.write(j)
                f.write(' ')
```

```
53.             f.write('\n')
```

jieba 的词汇表中并没有收录很多金庸的武侠小说这种特定环境下的很多专有名词，包括一些重要人物的名称，一些重要的武功等。这里整理了三份 txt 文本，分别记录了武侠小说中的人物名称、武功名称和门派名称，并把这些词汇添加到词汇表中。三个文档来源于参考文献 1.

总体步骤：替换无用字符、jieba 添加词汇、jieba 分词、去除停用词、保存为 txt 以便下一步训练。

2. Word2vec 训练

```
1.  from gensim.models import word2vec
2.
3.
4.  if __name__ == '__main__':
5.      sentences = word2vec.LineSentence('./sentences.txt')
6.
7.      model_cbow = word2vec.Word2Vec(sentences, sg=0, vector_size=100, window=
    5, min_count=5, workers=3, epochs=20)
8.      model_skip_gram = word2vec.Word2Vec(sentences, sg=1, vector_size=100, wi
    ndow=5, min_count=5, workers=3, epochs=20)
9.      model_cbow.save('./cbow.model')
10.     model_skip_gram.save('./skip_gram.model')
```

将上个步骤中获得的分词好的文本集送入 Word2vec 模型中训练，分别训练 CBOW 模型和 Skip-gram 模型，保存模型。

3. 模型测试与分析

```
1.  model_cbow = word2vec.Word2Vec.load('./cbow.model')
2.  model_skip_gram = word2vec.Word2Vec.load('./skip_gram.model')
3.
4.
5.  print(model_cbow.wv.most_similar('杨过', topn=10))
6.  print(model_cbow.wv.most_similar('郭靖', topn=10))
7.  print(model_cbow.wv.most_similar('韦小宝', topn=10))
8.  print(model_cbow.wv.most_similar('峨嵋派', topn=10))
9.  print(model_cbow.wv.most_similar('六脉神剑', topn=10))
10. print(model_cbow.wv.most_similar('降龙十八掌', topn=10))
11.
12. print('\n')
```

```
13.
14. print(model_skip_gram.wv.most_similar('杨过', topn=10))
15. print(model_skip_gram.wv.most_similar('郭靖', topn=10))
16. print(model_skip_gram.wv.most_similar('韦小宝', topn=10))
17. print(model_skip_gram.wv.most_similar('峨嵋派', topn=10))
18. print(model_skip_gram.wv.most_similar('六脉神剑', topn=10))
19. print(model_skip_gram.wv.most_similar('降龙十八掌', topn=10))
```

读取模型，通过模型给出与指定词最相关的 10 个词。

4. 结果分析

实验结果如下：

```
1. [('小龙女', 0.7538228631019592), ('郭襄', 0.6987199783325195), ('郭芙
   ', 0.6697977185249329), ('黄蓉', 0.6268817782402039), ('李莫愁
   ', 0.623081624507904), ('金轮法王', 0.6192827224731445), ('法王
   ', 0.5991382002830505), ('绿萼', 0.5981455445289612), ('陆无双
   ', 0.5856004953384399), ('郭靖', 0.5811579823493958)]
2. [('黄蓉', 0.7089859843254089), ('黄药师', 0.6316888928413391), ('欧阳锋
   ', 0.6255456805229187), ('柯镇恶', 0.5915210843086243), ('洪七公
   ', 0.5835084915161133), ('杨过', 0.581157922744751), ('欧阳克
   ', 0.5721482634544373), ('朱聪', 0.5529967546463013), ('周伯通
   ', 0.5355632305145264), ('华筝', 0.5353147387504578)]
3. [('康熙', 0.6417772173881531), ('郑克爽', 0.6175757050514221), ('双儿
   ', 0.6164399981498718), ('施琅', 0.6095462441444397), ('索额图
   ', 0.5919697284698486), ('小桂子', 0.572338342666626), ('海老公
   ', 0.5632778406143188), ('多隆', 0.5613011717796326), ('费要多罗
   ', 0.5562957525253296), ('阿珂', 0.5498335957527161)]
4. [('华山派', 0.7090849876403809), ('峨嵋', 0.7011258602142334), ('本派
   ', 0.6537455916404724), ('静玄', 0.6270795464515686), ('灭绝师太
   ', 0.624752938747406), ('恒山派', 0.6123185157775879), ('武当派
   ', 0.6106323599815369), ('魔教', 0.6035714745521545), ('五岳剑派
   ', 0.5973027348518372), ('静玄师太', 0.5779914259910583)]
5. [('一阳指', 0.7068790197372437), ('秘奥', 0.7060996294021606), ('独孤九剑
   ', 0.7055060267448425), ('大理段氏', 0.6884140372276306), ('要诀
   ', 0.6814315319061279), ('拳理', 0.6761385798454285), ('胡家刀法
   ', 0.6697395443916321), ('七伤拳', 0.6606950163841248), ('指法
   ', 0.6581071615219116), ('龙爪手', 0.6503828167915344)]
6. [('亢龙有悔', 0.7802537083625793), ('打狗棒法', 0.7689022421836853), ('棒法
   ', 0.7414721250534058), ('掌法', 0.7351680994033813), ('太极拳
   ', 0.7242019772529602), ('空明拳', 0.7227168083190918), ('神剑掌
   ', 0.722288966178894), ('八极拳', 0.7177032232284546), ('十八掌
   ', 0.716754674911499), ('胡家刀法', 0.7104465365409851)]
```

```
7.
8.
9.  [('小龙女', 0.8605223298072815), ('黄蓉', 0.7795921564102173), ('郭芙
    ', 0.7571137547492981), ('李莫愁', 0.754122257232666), ('郭靖
    ', 0.7321240901947021), ('陆无双', 0.7186410427093506), ('过儿
    ', 0.6898508667945862), ('金轮法王', 0.6866841912269592), ('法王
    ', 0.6645865440368652), ('黄药师', 0.6601234078407288)]
10. [('黄蓉', 0.8349590301513672), ('杨过', 0.7321240305900574), ('欧阳锋
    ', 0.7017830610275269), ('黄药师', 0.6849286556243896), ('蓉儿
    ', 0.6812700629234314), ('柯镇恶', 0.6672270894050598), ('靖哥哥
    ', 0.65334153175354), ('拖雷', 0.6170433163642883), ('杨康
    ', 0.6168296933174133), ('黄蓉笑', 0.6093813180923462)]
11. [('康熙', 0.7477850317955017), ('双儿', 0.7131544351577759), ('茅十八
    ', 0.6775522232055664), ('郑克爽', 0.670400857925415), ('阿珂
    ', 0.6569821834564209), ('皇上', 0.647208571434021), ('小桂子
    ', 0.6377466320991516), ('珂', 0.6318269968032837), ('索额图
    ', 0.6316999793052673), ('多隆', 0.6307588219642639)]
12. [('灭绝师太', 0.7487694025039673), ('周芷若', 0.675719141960144), ('峨嵋
    ', 0.6750054359436035), ('武当派', 0.6647878289222717), ('俞莲舟
    ', 0.6070315837860107), ('本派', 0.6053275465965271), ('殷梨亭
    ', 0.5965154767036438), ('丁敏君', 0.5918353199958801), ('宋青书
    ', 0.5899598002433777), ('金花婆婆', 0.588134229183197)]
13. [('大理段氏', 0.6518537402153015), ('枯荣', 0.6508990526199341), ('一阳指
    ', 0.6414085030555725), ('天龙', 0.6358145475387573), ('鸠摩智
    ', 0.6237537860870361), ('慕容先生', 0.6094193458557129), ('商阳剑
    ', 0.5913573503494263), ('火焰刀', 0.5902565717697144), ('本因
    ', 0.5758730173110962), ('剑气', 0.5710780620574951)]
14. [('打狗棒法', 0.7120189070701599), ('十八掌', 0.7012403011322021), ('掌法
    ', 0.6917229294776917), ('亢龙有悔', 0.6759560108184814), ('六七招
    ', 0.6305197477340698), ('逍遥游拳法', 0.6280497312545776), ('掌
    ', 0.6258848905563354), ('偏花', 0.6251261830329895), ('棒法
    ', 0.6250163912773132), ('十五招', 0.6247066855430603)]
```

针对两种模型，给出六个测试词汇，分别为三个人名、一个帮派名和两个招法名。首先，对于人名，模型给出的相关结果也均为人名或指代名（如皇上等），对于帮派名，CBOW 模型给出了一部分帮派名和一部分与峨嵋派紧密相关的人名（如灭绝师太、静谧师太），而 Skip-gram 模型更倾向于给出了与峨嵋派紧密相关的人名（灭绝师太、周芷若等）。对于招法名，两个模型也均给出了接近的招法名。

针对人物关系的提取，我认为本实验中 CBOW 模型完成的好一些，针对“杨过”，CBOW 提取了郭襄，郭芙，而在 Skip-gram 模型中没有体现，但是这二人

与杨过有着很密切的关系。综合来看，CBOW 可能更重视整体的关联，而 Skip_gram 看重局部的联系。

四、参考文献

1. https://blog.csdn.net/weixin_50891266/article/details/116750204
2. https://blog.csdn.net/weixin_44966965/article/details/124732760