浙江大学

本科实验报告

课程名称：自然语言处理

姓　　名：张溢弛

学　　院：计算机科学与技术学院

专　　业：软件工程 1801

学　　号：3180103772

指导教师：汤斯亮

2021 年 5 月 21 日

浙江大学 实验报告

课程名称：自然语言处理　　实验类型：综合型

实验项目名称：实验一：词向量

学生姓名：张溢弛　专业：软件工程 1801　学号：3180103772

同组学生姓名：独立完成　指导老师：汤斯亮

实验地点：玉泉一舍 376　实验日期：2021 年 5 月 21 日

目录

一 项目介绍 **IV**

1.1 实验选题 IV

1.2 实验内容 IV

1.3 实验环境 IV

二 技术细节 **IV**

2.1 Word2Vec 的原理 IV

2.2 实验细节与关键代码 V

2.2.1 实验环境配置 V

2.2.2 语料库预处理 V

2.2.3 模型训练 VI

2.2.4 模型的简单使用 VI

2.2.5 模型的保存 VII

三 实验结果 **VII**

3.1 二维词向量的可视化 VII

3.2 词向量模型的简单使用 VIII

3.2.1 单词分类 VIII

3.2.2 中心词预测 IX

3.2.3 相似单词预测 IX

3.3 模型的比较 IX

3.3.1 词向量维度对模型的影响 IX

3.3.2 窗口大小对词向量的影响 X

3.3.3 最少出现次数对词向量的影响 X

四 总结思考 **XI**

一　项目介绍

1.1　实验选题

基于中文语料库的 Word2Vec 词向量模型训练

1.2　实验内容

基于课程提供的中文语料库 (包含 10000 个句子) 进行 Word2Vec 词向量的训练，可以调用 gensim 库或者 MindSpore 框架的开源代码实现，并探究不同的参数对词向量模型的作用，并进行词向量相关的简单实验。

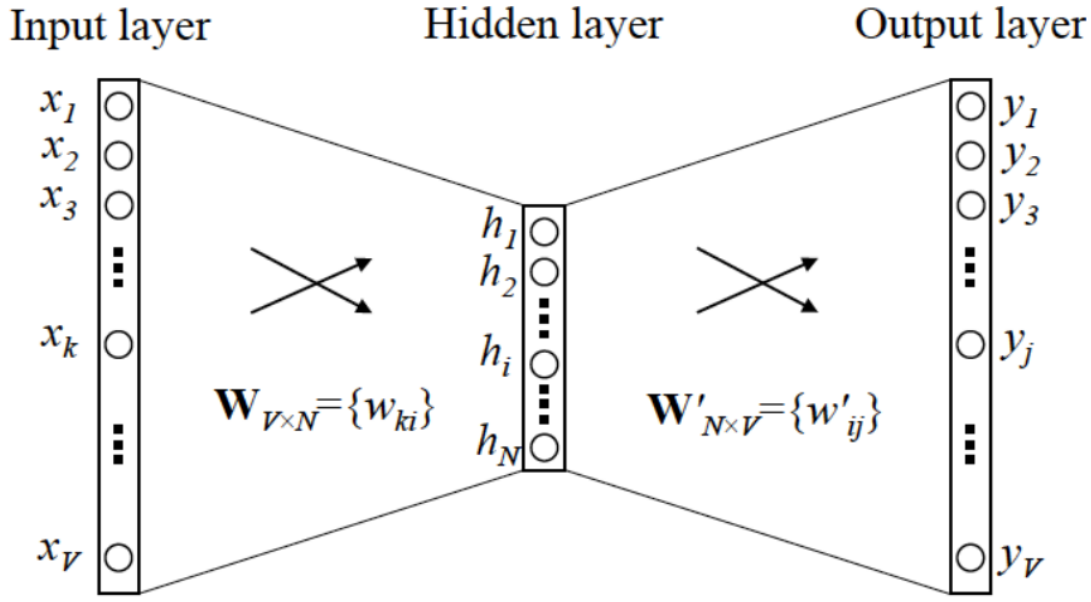
1.3　实验环境

- 操作系统：华为云 ModelArts 云平台
- 编程环境：Python3+Jupyter Notebook
- 使用的 Python 库：jieba(预处理)，gensim(训练模型)，matplotlib(可视化) 等

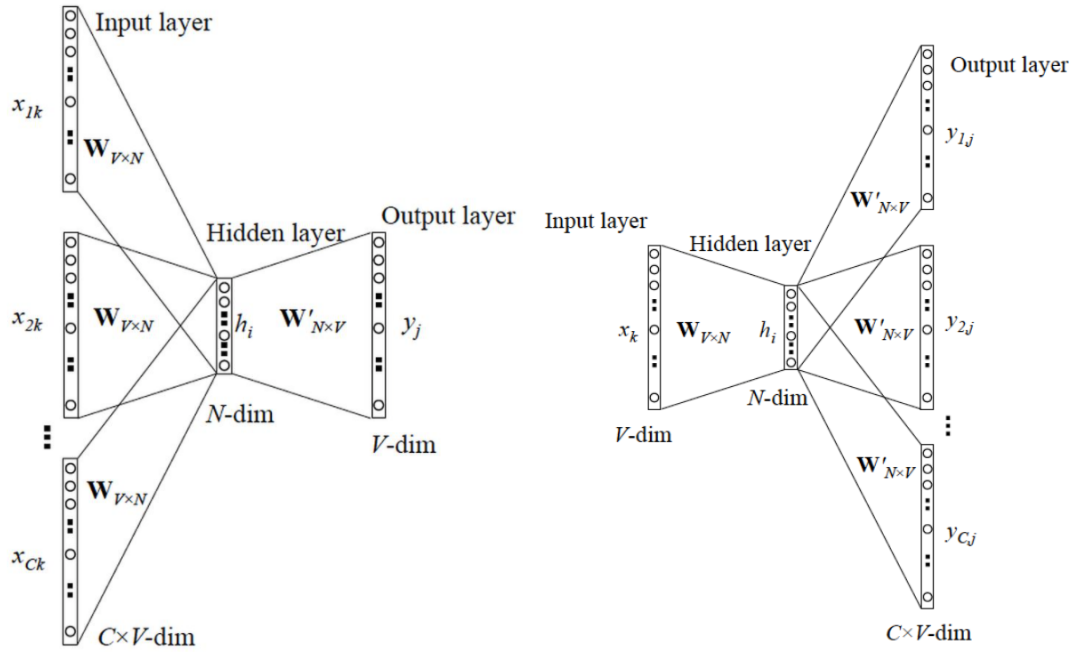
二　技术细节

2.1　Word2Vec 的原理

Word2Vec 是一种非常经典的词向量训练的方式，有 CBOW 和 Skip-gram 两种模型，分别在不同的任务中使用，CBOW 可以使用上下文单词预测中心单词，而 Skip-gram 可以通过中心单词预测上下文的单词，其本质是一个两层的神经网络：



而 CBOW 模型和 Skip-gram 模型更一般的形式如下图所示：



以 CBOW 模型为例，我们需要把上下文单词对应的向量加权之后输入到隐层中，再由隐层通过一个全连接层和 softmax 函数映射成最终的输出结果 (一个概率分布)，然后我们就可以根据概率分布来推测最合适的中心单词，要学习的参数主要是两个权重矩阵 W，但是本次实验中我们通过调用 gensim 库帮助我们非常方便地训练词向量，并将重点放在对词向量模型的探索上。

2.2 实验细节与关键代码

本次实验的细节和关键的代码主要分为如下几个部分，其中实验结果将在第三部分具体给出。

2.2.1 实验环境配置

注册华为云平台的账号，由于代金券还没有到账因此自费充值来使用 ModelArts 平台，按照教程中创建好 OBS 桶并在 ModelArts 中配置好实验环境之后开始实验，其中数据集需要通过 moxing 库从 OBS 桶中复制到云平台当前的工作目录下。

2.2.2 语料库预处理

在训练 Word2Vec 词向量模型之前需要对语料库进行一定的预处理，将语料库转化成一个分词构成的二维数组，因为语料库是中文的，因此我使用了 jieba 库进行分词操作，而语料中含有大量无关单词和标点符号，我在使用 jieba 分词之前先用正则表达式对其进行过滤，然后再使用 jieba 库进行中文预料的分词，最后再使用一个网上开源的中文 stop words 数据集 (可以带代码目录下找到)，将分词中的所有 stop words 去掉，来提高词向量训练的

效果。整个语料库处理过程通过一个函数 sentence_preprocess 来实现，该函数的代码如下所示 (也可以在提交的代码文件中找到)：

```
import jieba
import re


def sentence_preprocess(sentence):
    pattern = re.compile(r'[\u4e00-\u9fa5]+')
    # 去掉所有不是中文的词
    sentence = re.sub(r"[^\u4e00-\u9fff]", " ", sentence)
    sentence = re.sub(r"\s{2,}", " ", sentence)
    participle = jieba.lcut(sentence)
    word_without_stopwords = []
    # 去除stop word
    for word in participle:
        if word not in stop_words and word != ' ':
            word_without_stopwords.append(word)
    return word_without_stopwords
```

代码中的 u4e00-u9fff 就是将所有非中文的内容进行过滤，这是因为汉字的 UTF-8 编码刚好在这个范围内。然后我们对整个语料库中的每个句子使用这个函数进行预处理，最终可以得到预处理之后的分词列表，如下图所示：

```
Building prefix dict from the default dictionary ...
DEBUG:jieba:Building prefix dict from the default dictionary ...
Dumping model to file cache /tmp/jieba.cache
DEBUG:jieba:Dumping model to file cache /tmp/jieba.cache
Loading model cost 1.104 seconds.
DEBUG:jieba:Loading model cost 1.104 seconds.
Prefix dict has been built succesfully.
DEBUG:jieba:Prefix dict has been built succesfully.
```

```
[['讲', '孔子', '后人', '故事', '一个', '老', '领导', '回到', '家乡', '儿子', '感情', '不', '贪财', '孙子', '孔为', '和睦', '老', '领导', '弟弟', '魏宗万', '马车', '有个', '洋妞', '大概', '考察', '民俗', '家', '过年', '孔为', '总想', '出国', '爷爷', '教育', '一家人', '和解', '顺便', '问', '另一类', '电影', '北京', '青年电影制片厂', '中', '越战', '背景', '军人', '介绍', '一个', '对象', '去', '相亲', '女方', '军队', '医院', '护士', '犹豫不决', '回忆', '战场', '上', '负伤', '男友', '好像', '还', '没', '死', '男方', '理解', '归队'],
['不至于', '离开', '破', '公司', '课题', '做', '谢谢', '关心', '昨天晚上', '睡', '很', '好', '想', '好', '见机行事', '拿到', '相关', '做', '论文', '材料', '马上', '辞职', '说不定', '还要', '各为', '帮出', '出', '找', '工作', '主意', '学', '通信', '哈尔滨工程大学', '研究生', '不想', '碌碌无为', '做', '设计', '才', '先', '谢谢', '本人', '语文', '不好', '没加', '标点', '辛苦', '看不懂']]
```

预处理步骤至此完成。

2.2.3 模型训练

本次实验我直接调用了 gensim.models 中的 Word2Vec 库进行，可以用 Word2Vec(sentences, size=2, window=5, min_count=5, workers=5) 的方式直接调用库函数训练出 Word2Vec 模型，只需要指定好词向量的维数，窗口和最少出现次数等超参数的值就可以。

而在本次实验中，我通过调整超参数的值，训练出了多个 Word2Vec 模型，并对其性能进行了比较，这一部分的内容主要在实验结果中展示。

2.2.4 模型的简单使用

实验中我还使用训练好的模型进行了了一些简单的 task 的测试，比如词向量的可视化、单词分类、相似度预测、最高匹配度单词检索等等，这里主要是调用了 gensim 中 Word2Vec 模型的一些库函数比如 wv.doesnt_match 进行单词分类，wv.most_similar 进行相似单词检

索，wv.vectors 获取词向量并使用 matplotlib 库对二维词向量进行可视化展示，并对不同参数训练出来的模型进行了比较分析，得出一定的结论。

2.2.5 模型的保存

在探索了超参数对于 Word2Vec 模型的影响之后，我选择了较优的参数训练出了一个性能比较好的词向量模型，并将其进行了保存，如下图所示：

```
new_model.save('word2vec.model')
```

```
INFO:gensim.utils:saving Word2Vec object under word2vec.model, separately None
INFO:gensim.utils:not storing attribute vectors_norm
INFO:gensim.utils:not storing attribute cum_table
INFO:gensim.utils:saved word2vec.model
```

```
! ls -al
```

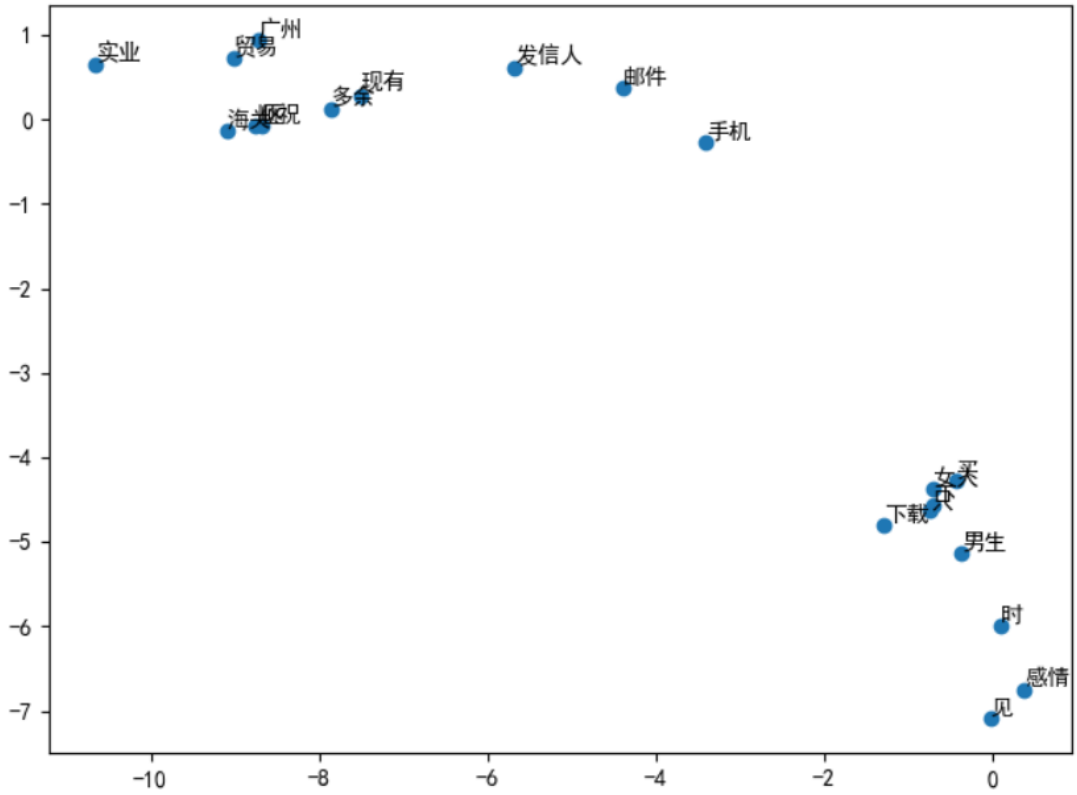
```
total 9276
drwxr-x---  3 ma-user ma-group      72 May 21 22:40 .
drwxr-x--- 21 ma-user ma-group    4096 May 21 21:23 ..
-rw-r-----  1 ma-user ma-group 3785340 May 21 21:24 data
drwxr-xr-x  2 ma-user ma-group       6 May 21 21:20 .script
-rw-r-----  1 ma-user ma-group    6407 May 21 21:24 stopwords
-rw-r-----  1 ma-user ma-group 5693589 May 21 22:40 word2vec.model
```

三 实验结果

本次实验中训练 Word2Vec 模型得到的一些结果如下所示。

3.1 二维词向量的可视化

在本次实验中，我首先训练了一个词向量维数为 2 的 Word2Vec 模型 (即指定 size=2)，并将其使用 matplotlib 库进行可视化，具体的代码可以在因为单词太多所以我只从中选取了 15 个单词，最终得到的结果如下：



我们可以从图中分析出 Word2Vec 训练的词向量中，彼此靠近的词向量存在一定的逻辑关系，比如广州-贸易-海关，发信人-邮件-手机，女人-男生等等，说明词向量在训练的过程中会将类型相似或者有一定逻辑联系的单词编码到邻近的位置中，这也表明我们的模型训练是正确的。

3.2 词向量模型的简单使用

我另外训练了一个词向量模型，采用 100 维的词向量模型来进行一些简单的词向量任务。

3.2.1 单词分类

使用 wv.doesnt_match 找出列表中不同类型的单词：

```
# 单词分类
test_list = ['虚伪', '善良', '勇敢', '美妙绝伦']
print(model_test.wv.doesnt_match(test_list))
```

```
INFO:gensim.models.keyedvectors:precomputing L2-norms of word weight vectors
```

```
虚伪
```

这里的 4 个词语中有 3 个是褒义词，1 个是贬义词，我训练的 100 维词向量模型准确的分辨出了贬义词。

3.2.2 中心词预测

使用 predict_output_word 在另一个训练好的 CBOW 模型上进行中心词的预测 (Word2Vec 训练时需要制定特殊的参数才能得到 CBOW 模型)，结果如下：

```
# 中心词预测
context = ['公司', '化工', '总部']
print(model_test.predict_output_word(context, topn=5))
```

```
[('代理', 0.0013340871), ('提供', 0.00083328836), ('贵', 0.00083175075), ('企业', 0.0008305235), ('全国', 0.00079702685)]
```

可以看出中心词的预测还是存在一定的逻辑性的。

3.2.3 相似单词预测

预测与输入单词相似度最高的单词，我们使用上面同一个模型的预测结果如下：

```
# 相似度预测
print(model_test.wv.most_similar_cosmul('高校', topn=10))
```

```
[('名校', 0.9967678785324097), ('背景', 0.9961206912994385), ('调干', 0.9961103796958923), ('毕业生', 0.9956440329551697), ('创业者', 0.9952207803726196), ('计算机系', 0.9949651956558228), ('自动化', 0.9948566555976868), ('组织', 0.9948160648345947), ('计算机', 0.9947742223739624), ('大专', 0.9947642087936401)]
```

可以看到单词预测结果和输入的单词存在比较强的逻辑联系

3.3 模型的比较

我通过调整模型的超参数训练得到了一系列不同的 Word2Vec 模型，并对其性能进行了一定的比较。

3.3.1 词向量维度对模型的影响

我训练了维度分别为 2，10 和 100 的三个模型，并使用相似单词预测来对比研究其性能：

```
# 比较三种模型的同类词推理结果
pred1 = model_2d.wv.most_similar('公司', topn=10)
print(pred1)
pred2 = model_10d.wv.most_similar('公司', topn=10)
print(pred2)
pred3 = model_100d.wv.most_similar('公司', topn=10)
print(pred3)
```

```
INFO:gensim.models.keyedvectors:precomputing L2-norms of word weight vectors
INFO:gensim.models.keyedvectors:precomputing L2-norms of word weight vectors
[('打扰', 0.9999999403953552), ('普通发票', 0.9999999403953552), ('加一', 0.9999998211860657), ('地税', 0.9999997615814209), ('代开', 0.9999995827674866), ('侯', 0.9999989867210388), ('票', 0.9999983310699463), ('实业', 0.9999982714653015), ('书刊', 0.9999982118606567), ('发票', 0.9999979138374329)]
[('工厂', 0.9768513441085815), ('收', 0.9732964634895325), ('现我司', 0.9732546806335449), ('我司', 0.9628394246101379), ('更能', 0.9616590738296509), ('开到', 0.9615283012390137), ('利益', 0.9568119049072266), ('为贵', 0.9560458064079285), ('长期', 0.9544855356216431), ('外贸生意', 0.9528414011001587)]
[('开到', 0.9578976035118103), ('余额', 0.9477720260620117), ('托运', 0.9474736452102661), ('更能', 0.9459636211395264), ('工厂', 0.9450891613960266), ('请速', 0.9448772072792053), ('收', 0.9425256848335266), ('为贵', 0.9355312585830688), ('税金', 0.9315922856330872), ('可为', 0.9288512468338013)]
```

```
pred1 = model_2d.wv.most_similar('高校', topn=10)
print(pred1)
pred2 = model_10d.wv.most_similar('高校', topn=10)
print(pred2)
pred3 = model_100d.wv.most_similar('高校', topn=10)
print(pred3)
```

```
[('一篇', 1.0), ('网电', 1.0), ('导航', 1.0), ('结论', 1.0), ('细节', 1.0), ('好不好', 0.9999999403953552), ('垃圾', 0.9999999403953552), ('人大代表', 0.9999999403953552), ('体谅', 0.9999999403953552), ('前台', 0.9999998807907104)]
[('组织', 0.9925017356872559), ('以上学历', 0.9915351271629333), ('打造', 0.9905723929405212), ('攻读', 0.9894732236862183), ('招', 0.9888353394985199), ('大学毕业', 0.9881454110145569), ('压制', 0.9879406094551086), ('所幸', 0.9876693487167358), ('中科院', 0.9873824715614319), ('论文', 0.9873535037040711)]
[('自动化', 0.9933631420135498), ('本科学历', 0.9910752773284912), ('经济', 0.9901901483535767), ('化学', 0.9886677861213684), ('市场', 0.9884859919548035), ('计算机系', 0.9879967570304871), ('大专', 0.9872817397117615), ('积蓄', 0.9870378971099854), ('起订', 0.9863682389259338), ('攻读', 0.9862586855888367)]
```

我们从两个测试的对比中判断，维度比较高的词向量更容易捕捉到单词之间的关系和特征，

维数比较高的词向量预测的效果比较好，可以更好地表示单词的特征。当然如果设置的过大可能也会导致词向量在嵌入空间中的表示太稀疏而表达能力下降。

3.3.2 窗口大小对词向量的影响

通过改变模型超参数 window 可以控制 Word2Vec 词向量模型中滑动窗口的大小，我们将向量维数固定在 100 维，并用 2，5，10 分别作为窗口大小来训练 3 个词向量模型，并比较其性能：

```
pred2 = model_100d_2.wv.most_similar('公司', topn=10)
print("窗口大小为2", pred2)
pred1 = model_100d.wv.most_similar('公司', topn=10)
print("窗口大小为5", pred1)
pred3 = model_100d_3.wv.most_similar('公司', topn=10)
print("窗口大小为10", pred3)
```

```
窗口大小为2 [('可为', 0.9153035879135132), ('集团', 0.8982906341552734), ('收', 0.890457272529602), ('我司', 0.8898220062255859), ('可代', 0.8817199468612671), ('利益', 0.8766655325889587), ('谋利', 0.8759021162986755), ('现我司', 0.873476505279541), ('为贵', 0.8729824423789978), ('进项', 0.8702268600463867)]
窗口大小为5 [('开到', 0.9578976035118103), ('余额', 0.9477720260620117), ('托运', 0.9474736452102661), ('更能', 0.9459636211395264), ('工厂', 0.9450891613960266), ('请速', 0.9448772072792053), ('收', 0.9425256848335266), ('为贵', 0.9355312585830688), ('税金', 0.9315922856330872), ('可为', 0.9288512468338013)]
窗口大小为10 [('余额', 0.9698547124862671), ('长期', 0.9696630239486694), ('开到', 0.9668271541595459), ('更能', 0.9632536172866821), ('外开', 0.9597975015640259), ('票向', 0.9594708681106567), ('均', 0.9548945426940918), ('现我司', 0.9442549347877502), ('优惠价格', 0.9435784220695496), ('现', 0.9433513879776001)]
```

```
pred1 = model_100d_2.wv.most_similar('高校', topn=10)
print("窗口大小为2\n", pred1)
pred2 = model_100d.wv.most_similar('高校', topn=10)
print("窗口大小为5\n", pred2)
pred3 = model_100d_3.wv.most_similar('高校', topn=10)
print("窗口大小为10\n", pred3)
```

```
窗口大小为2
 [('积累', 0.9971309900283813), ('向上', 0.997122526168232), ('方向', 0.9968839287757874), ('公园', 0.9964501857757568), ('治疗', 0.9963911771774292), ('京', 0.9960951805114746), ('高中毕业', 0.9960571527481079), ('高三', 0.9959682822227478), ('不妥', 0.995711922645568), ('手板', 0.9956993460655212)]
窗口大小为5
 [('自动化', 0.9933631420135498), ('本科学历', 0.9910752773284912), ('经济', 0.9901901483535767), ('化学', 0.9886677861213684), ('市场', 0.9884859919548035), ('计算机系', 0.9879967570304871), ('大专', 0.9872817397117615), ('积蓄', 0.9870378971099854), ('起订', 0.9863682389259338), ('攻读', 0.9862586855888367)]
窗口大小为10
 [('试验', 0.984568178653717), ('管理者', 0.983007550239563), ('博', 0.982240617275238), ('经济学', 0.9796931147575378), ('规划', 0.9794517755508423), ('背景', 0.9790149927139282), ('职责', 0.9781290292739868), ('本科毕业', 0.977015495300293), ('本科学历', 0.9765214920043945), ('所属', 0.9762600660324097)]
```

可以分析出窗口越大，越可以捕捉到更多的单词之间的上下文关系，预测的结果也会有相对应的提升。

3.3.3 最少出现次数对词向量的影响

最少出现次数通过设置 min_count 参数来调整，在语料库中出现次数低于这个数的单词将不会被放到模型中进行训练，也就是说这一参数的设置可以过滤掉出现频率比较低的词语，训练出出现频率较高的单词的词向量，调整这个参数带来的最直观的感受就是模型训练的输出日志中，训练的单词量随着 min_count 调整变大而逐渐变小，最后我们通过模型的比较得到的结果是：

```
pred2 = model_100d_4.wv.most_similar('高校', topn=10)
print("最少出现2次\n", pred2)
pred1 = model_100d.wv.most_similar('高校', topn=10)
print("最少出现5次\n", pred1)
pred3 = model_100d_5.wv.most_similar('高校', topn=10)
print("最少出现10次\n", pred3)
```

```
最少出现2次
 [('背景', 0.9968065023422241), ('策', 0.9965357184410095), ('出租', 0.9958497285842896), ('出谋画', 0.9953406453132629), ('诸多', 0.9953097105026245), ('增加', 0.9952226281166077), ('编程', 0.9950042963027954), ('源码', 0.9948739409446716), ('以上学历', 0.9947587847709656), ('方向', 0.9943267703056335)]
最少出现5次
 [('自动化', 0.9933631420135498), ('本科学历', 0.9910752773284912), ('经济', 0.9901901483535767), ('化学', 0.9886677861213684), ('市场', 0.9884859919548035), ('计算机系', 0.9879967570304871), ('大专', 0.9872817397117615), ('积蓄', 0.9870378971099854), ('起订', 0.9863682389259338), ('攻读', 0.9862586855888367)]
最少出现10次
 [('方向', 0.9855301380157471), ('大专', 0.9836357831954956), ('中科院', 0.9836231470108032), ('休闲', 0.9815608859062195), ('水平', 0.9814247488975525), ('软件开发', 0.981342613697052), ('背景', 0.9812928438186646), ('成本', 0.9806684851646423), ('院士', 0.9800575971603394), ('座', 0.9796621799468994)]
```

我们从结果中可以判断 min_count 不宜太小也不宜太大，适中的 min_count 可以学习到更多的上下文信息 (设置的太大可能就会导致可训练的单词太少，信息丢失)

四　总结思考

本次实验通过调用 gensim 训练词向量模型，使我初步学习了华为云人工智能平台的使用，并进行了文本预处理、词向量参数调节、词向量的可视化、词向量基本功能的简单使用等环节的实践，加深了我对词向量和 Word2Vec 模型的理解。虽然因为语料库比较小且是中文文本，句子关系也比较杂乱，模型的性能并不好，但我还是从实验中学习到了很多。

我从实验的过程中意识到，Word2Vec 的训练是一种无监督学习，实验中没有使用明确的量化评价标准来评价模型的好坏，很多时候评价不同参数的模型之间的优劣只能通过具有较强主观性的判断 (直接看生成的结果哪个更匹配更具有逻辑性) 来进行，而参数的调节作为一个比较玄学的问题，也有待进一步的学习和实践。

参考文献

[1] http://web.stanford.edu/class/cs224n/index.html

[2] https://radimrehurek.com/gensim/auto_examples/tutorials/run_-word2vec.html#sphx-glr-auto-examples-tutorials-run-word2vec-py

[3] http://gensim.apachecn.org/#/