Report of Deep Learning for Natural Langauge Processing

ZY2303217 郑茗畅
zy2303217@buaa.edu.cn

Abstract

本研究使用金庸的十六部经典武侠小说作为语料库，利用 Word2Vec 来训练词向量，通过计算词向量之间的语意距离、相关词向量的 K-means 聚类、某些段落直接的语意关联等方法来验证词向量的有效性。

Introduction

I1: 词向量

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。为了描述每个词汇的特征， Hinton 在 1986 年提出了使用词向量来表示词。词向量是自然语言处理（NLP）领域中的一个核心概念，它将词汇表中的每个单词映射到一个固定维度的实数向量。这样的向量不仅包含了单词的语义信息，而且还能够反映出单词之间的语义关系。

词向量有两种表示形式，一种是One-hot编码，另一种是分布式表示。

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）。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。

I2: 神经语言模型

神经语言模型是自然语言处理中的典型方法，其主要功能是根据当前单词进行相关单词的预测。神经语言模型的输出为其所知单词的概率评分，通常使用百分比表示。模型在经过训练后会生成表示所有单词的映射矩阵，在预测中，需要在该映射矩阵中查询输入单词，然后计算对应的预测值。在本次实验中，采用了Word2Vec神经语言模型这一流行的词向量训练方法，来生成词向量。

Word2Vec是由Google在2013年提出的一种词嵌入技术。它通过训练神经网络模型，将词语表示成向量，使得具有相似语义的词在向量空间中相互靠近。其模型结构如图1所示。

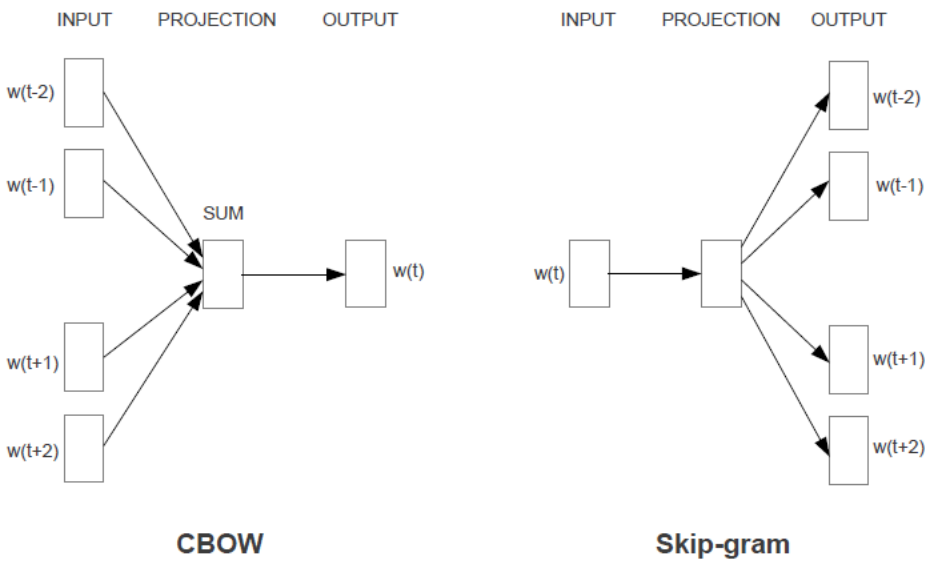


图 1 Word2Vec模型结构

Word2Vec有两种主要的模型架构：连续词袋模型（CBOW）和跳字模型（Skip-Gram）。CBOW通过上下文词预测中心词，而Skip-Gram通过中心词预测上下文词。CBOW模型结构和Skip-gram模型结构分别如图2和图3所示。两种模型均是通过权重矩阵先将输入层的上下文或当前词的词向量映射至隐藏层向量，再用另一个权重矩阵将隐藏层向量映射至输出层词向量从而得到预测词语结果。

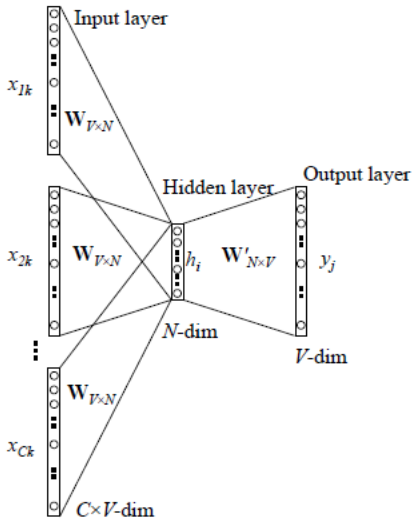


图 2 CBOW模型结构

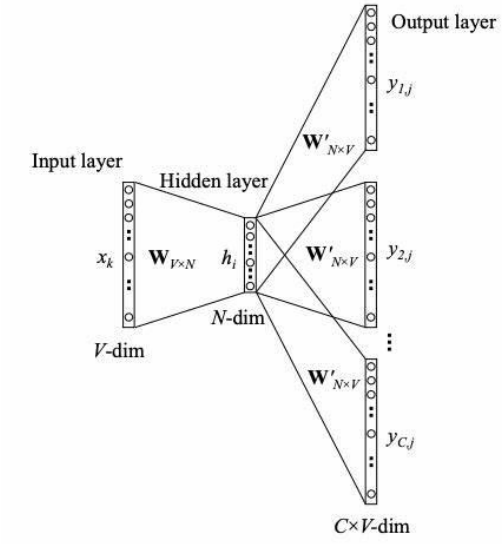


图 3 Skip-gram模型结构

在训练过程中Word2Vec使用大量文本数据进行无监督学习，通过最小化预测误差，不断调整词向量，使得具有相似语境的词语向量越来越接近。为了提高训练效率，Word2Vec引入了负采样（Negative Sampling）和分层Softmax（Hierarchical Softmax）技术，这些技术可以显著减少计算量。负采样通过计算中心词与背景词不同时出现的概率降低计算量；层次Softmax使用Huffman 树代替从隐藏层到输出层softmax 的映射，使用二元逻辑回归的方式进行判别，将计算量从V 降低到$\log_2 V$，且高频词靠近根节点，高频词的查询时间更短。

Methodology

M1: Word2Vec

算法流程如下：

1. 语料库处理
   1） 使用 16 篇金庸小说作为语料数据库；
   2） 删除部分文件中的无用信息；
   3） 删除无意义符号，包括标点符号和空白符号等；
   4） 使用 jieba 算法库进行分词；
   5） 根据 cn_stopwords.txt 删除中文停词；
   6） 对语料库整体以 100 个词汇为单位进行分句，并保存为 txt 文件，其中句子间用\n 分割，词汇间用空格分割。
2. 模型训练
   1） 读取保存处理后语料的 txt 文件；
   2） 使用 gensim 提供的 word2vec 训练语言模型，包括 CBOW 和 Skip-gram 两种模型架构；
   3） 保存语言模型。
3. 验证词向量有效性
   1） 计算词向量之间的语义距离；
   2） 使用 TSNE 将训练得到的模型词向量进行降维，并使用 K-Means 聚类算法对词语进行聚类，采用散点图进行效果展示，检验词向量；
   3） 计算段落的语意关联。

其中训练Word2Vec模型时具体参数如下：

- min_count = 5，可以对字典做截断，如果词频少于 5 的单词会被丢弃；
- window = 5，表示当前词与预测词在一个句子中的最大距离是 5；
- vector_size=100，指特征向量的维度，大的 size 需要更多的训练数据；
- sg=0/1，用于设置训练算法，其中 0 对应 CBOW 算法，1 代表 skip-gram 算法；
- workers=multiprocessing.cpu_count()，线程数；
- epochs=50，训练迭代轮数，过大会导致训练时间过长；
- hs=1, 采用 hierarchica·softmax 技巧。

Experimental Studies

E1: Word2Vec

1. 语料库处理

代码：

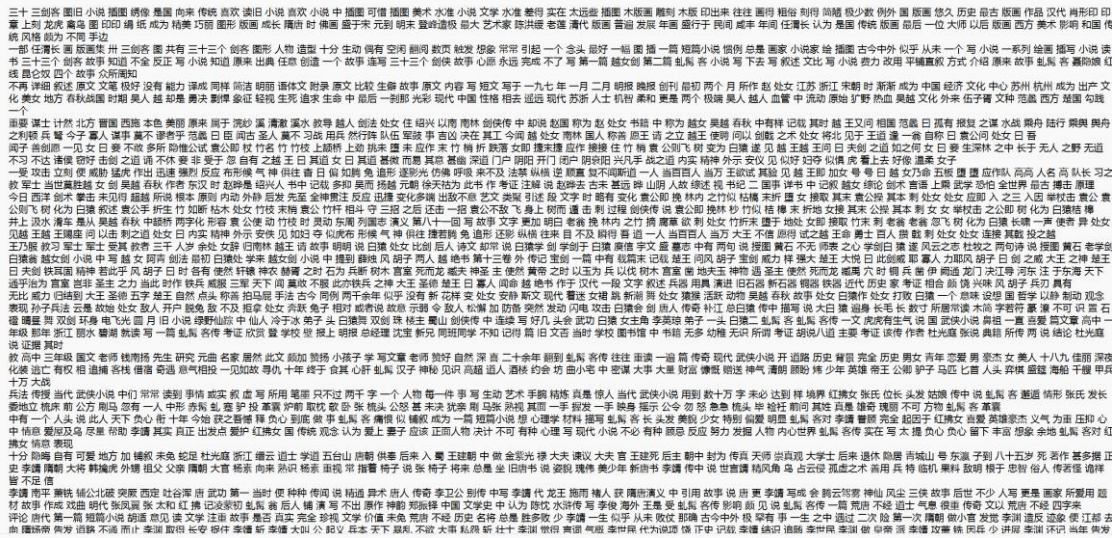
```
def preprocess_text(novel_name):
    with open(novel_name, 'r', encoding='ANSI') as f:
        con = f.read()
        ad = ['本书来自 www.cr173.com 免费 txt 小说下载站\n 更多更新免费电子书请关注 www.cr173.com', '----〖新语丝电子文库(www.xys.org)〗', '新语丝电子文库','Last Updated: Saturday, November 16, 1996','免费小说',
          '\u3000', '\n', '\r','\t','。', '？', '！', '，', '；', '：', '、', '《', '》', '“', '”', '‘', '’', '［', '］', '....', '......','【','】','—','-',',
','.',':',';','?','!','[',']','(',')','{','}','<','>',
          '『', '』', '（', '）', '…', '「', '」', '\ue41b', '＜', '＞', '+', '\x1a', '\ue42b', '她', '他', '你', '我', '它', '这'] #去掉其中的一些无意义的词语
        for a in ad:
            con = con.replace(a, '')
        text = re.sub(r'[^\u4e00-\u9fa5]', '', con)
    return text
def load_corpus(folder_path):
    names= os.listdir(folder_path)
    corpus=[]
    for name in names:
        novel_name = folder_path + '\\' + name
        processed_text=preprocess_text(novel_name)
        corpus.append(processed_text)
    return corpus,names
def cut_text(text, stopwords):
    words = jieba.lcut(text)
    return [word for word in words if word not in stopwords and word.strip()]
# 生成数据
def dataset_produce():
    sent_len=100
    words=[]
    folder_path = r'D:\课程\研一下 NLP\work3\novels'  # 文本文件夹路径
    corpus,names=load_corpus(folder_path)
    with open(r'D:\课程\研一下 NLP\work3\stopwords.txt', 'r', encoding='utf8') as f:
        stopwords = set([word.strip('\n') for word in f.readlines()])

    processed_corpus = [cut_text(text, stopwords) for text in corpus]

    for i in range(16):
        names[i]=re.sub('.txt','',names[i])
        savepath='pickle/'+names[i]+".pickle"
        with open(savepath,'wb') as f:
            pickle.dump(processed_corpus[i],f)
        with open(savepath, 'rb') as f:
            data = pickle.load(f)
        words_len = len(data)
        sent_num = math.ceil(words_len/sent_len)
        for i in range(sent_num):
            if i == sent_num-1:
                tmp = data[i*sent_len:-1]
            else:
                tmp = data[i*sent_len:(i+1)*sent_len]
            words.append(tmp)
    with open(r'D:\课程\研一下 NLP\work3\data.txt', 'w', encoding='utf-8') as f:
        data_str = '\n'.join([' '.join(row) for row in words])
```

```
48.             f.write(data_str)
```

结果：



2. 模型训练

代码：

```
1.  sentences = list(word2vec.LineSentence('data.txt'))
2.  model = Word2Vec(sentences, hs=1, min_count=5, window=5,vector_size=10
    0, sg=0, workers=multiprocessing.cpu_count(), epochs=50)
3.  model.save("word2vec.model")
```

结果：

word2vec.model	2024/6/1 22:43	MODEL 文件	59,574 K
word2vec——skipgram.model	2024/6/2 20:50	MODEL 文件	59,574 K

3. 验证词向量有效性

代码：

```
1.  model = Word2Vec.load(r'D:\课程\研一下 NLP\work3\word2vec.model')
2.  # 词语聚类
3.  test_name = ['李文秀', '袁承志', '胡斐', '狄云', '韦小宝', '剑客', '郭靖
    ', '杨过'
4.              , '陈家洛', '萧峰', '石破天', '令狐冲', '胡斐', '张无忌', '袁冠
    南', '阿青','剑法','客栈']
5.
6.  for name in test_name:
7.      print(name)
8.      for result in model.wv.similar_by_word(name, topn=10):
9.          print(result[0], '{:.6f}'.format(result[1]))
10.     print('--------------------------')
11. # 语意距离
12. print(model.wv.similarity('杨过', '小龙女'))
13. print(model.wv.similarity('杨过', '东方'))
14. print(model.wv.similarity('华山派', '弟子'))
15. print(model.wv.similarity('华山派', '剑法'))
16. print(model.wv.similarity('刀光', '剑影'))
17. print(model.wv.similarity('猫', '狗'))
18. print(model.wv.similarity('牛肉', '剑影'))
19.
20. names = [result[0] for name in test_name for result in model.wv.simila
    r_by_word(name, topn=10)]
21. name_vectors = np.array([model.wv[name] for name in names])
22. tsne = TSNE()
```

```
embedding = tsne.fit_transform(name_vectors)
n = 6
label = KMeans(n).fit(embedding).labels_
plt.title('kmeans 聚类结果')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
for i in range(len(label)):
    if label[i] == 0:
        plt.plot(embedding[i][0], embedding[i][1], 'ro', )
    if label[i] == 1:
        plt.plot(embedding[i][0], embedding[i][1], 'go', )
    if label[i] == 2:
        plt.plot(embedding[i][0], embedding[i][1], 'yo', )
    if label[i] == 3:
        plt.plot(embedding[i][0], embedding[i][1], 'co', )
    if label[i] == 4:
        plt.plot(embedding[i][0], embedding[i][1], 'bo', )
    if label[i] == 5:
        plt.plot(embedding[i][0], embedding[i][1], 'mo', )
    plt.annotate(names[i], xy=(embedding[i][0], embedding[i][1]), xytext=(embedding[i][0]+0.1, embedding[i][1]+0.1))
plt.show()
plt.savefig('cluster.png')

sample_paragraph1_str = '黄蓉不答，拉著他手走到後院，四下瞧了瞧，这才说道：“你和过儿的对答，我在窗外都听见啦。他不怀好意，你知道麽？”郭靖吃了一惊，问道：“甚麽不怀好意？”黄蓉道：“我听他言中之意，早在疑心咱俩害死了他爹爹。”郭靖道：“他或许确有疑心，但我已答允将他父亲逝世的情由详细说给他知道。”黄蓉道：“你当真要毫不隐瞒的说给他听？”郭靖道：“他父亲死得这麽惨，我心中一直自责。杨康兄弟虽然误入歧途，但咱们也没好好劝他，没想法子挽救。”黄蓉哼了一声，道：“这样的人又有甚麽可救的？我只恨杀他不早，否则你那几位师父又何致命丧桃花岛上？”郭靖想到这桩恨事，不禁长长叹了口气。'
sample_paragraph2_str = '黄蓉道：“朱大哥叫芙儿来跟我说，这次过儿来到襄阳，神气中很透著点儿古怪，又说你和他同榻而眠。我担心有何意外，一直守在你窗下。我瞧还是别跟他睡在一房的好，须知人心难测，而他父亲……总是因为一掌拍在我肩头，这才中毒而死。”郭靖道：“那可不能说是你害死他的啊。”黄蓉道：“既然你我均有杀他之心，结果他也因我而死，那麽是否咱们亲自下手，也没多大分别。”郭靖沉思半晌，道：“你说得对。那麽我还是不跟他明言的为是。蓉儿，你累了半夜，快回房休息罢。过了今晚，明日我搬到军营中睡。”'
paragraph_vector1 = get_paragraph_vector(sample_paragraph1_str, model)

paragraph_vector2 = get_paragraph_vector(sample_paragraph2_str, model)

paragraph_similarity = cosine_similarity([paragraph_vector1], [paragraph_vector2])[0][0]
print(f"Semantic similarity between paragraphs: {paragraph_similarity}")

sample_paragraph1_str = '中国共产党已走过百年奋斗历程。我们党立志于中华民族千秋伟业，致力于人类和平与发展崇高事业，责任无比重大，使命无上光荣。全党同志务必不忘初心、牢记使命，务必谦虚谨慎、艰苦奋斗，务必敢于斗争、善于斗争，坚定历史自信，增强历史主动，谱写新时代中国特色社会主义更加绚丽的华章。'
sample_paragraph2_str = '十九大以来的五年，是极不寻常、极不平凡的五年。党中央统筹中华民族伟大复兴战略全局和世界百年未有之大变局，就党和国家事业发展作出重大战略部署，团结带领全党全军全国各族人民有效应对严峻复杂的国际形势和接踵而至的巨大风险挑战，以奋发有为的精神把新时代中国特色社会主义不断推向前进。'
paragraph_vector1 = get_paragraph_vector(sample_paragraph1_str, model)

paragraph_vector2 = get_paragraph_vector(sample_paragraph2_str, model)


paragraph_similarity = cosine_similarity([paragraph_vector1], [paragraph_vector2])[0][0]
print(f"Semantic similarity between paragraphs: {paragraph_similarity}")
```

结果：

1） 相关词汇

模型训练好后，读取训练好的CBOW模型和Skip-gram模型，然后指定某一个词，展示与该词最相关的10个词。指定的词包括：韦小宝、郭靖、杨过、张无忌、令狐冲、峨嵋派、屠龙刀、蛤蟆功、葵花宝典。其涵盖了人名、门派名、武器名、功夫名、重要物品名等。结果如下所示。

表 1 CBOW模型结果

Related words of 韦小宝	[('康熙', 0.835749), ('郑克爽', 0.746565), ('太后', 0.692785), ('小桂子', 0.685975), ('双儿', 0.684165)]
Related words of 郭靖	[('黄蓉', 0.879300), ('欧阳锋', 0.841761), ('黄药师', 0.830460), ('洪七公', 0.821380), ('欧阳克', 0.820416)]
Related words of 杨过	[('黄蓉', 0.837252), ('小龙女', 0.826198), ('李莫愁', 0.819642), ('郭靖', 0.792364), ('法王', 0.787311)]
Related words of 张无忌	[('周芷若', 0.858927), ('谢逊', 0.813616), ('赵敏', 0.809844), ('张翠山', 0.783148), ('金花婆婆', 0.744007)]
Related words of 令狐冲	[('岳不群', 0.861111), ('盈盈', 0.796409), ('林平之', 0.781905), ('仪琳', 0.780648), ('岳灵珊', 0.776891)]
Related words of 峨嵋派	[('本派', 0.628514), ('武当派', 0.599218), ('本门', 0.570535), ('明教', 0.564184), ('华山派', 0.553822)]
Related words of 屠龙刀	[('谢逊', 0.616127), ('宝刀', 0.592478), ('金花婆婆', 0.561927), ('倚天剑', 0.542334), ('俞莲舟', 0.534028)]
Related words of 蛤蟆功	[('欧阳锋', 0.514600), ('九阴真经', 0.505926), ('老毒物', 0.482781), ('西毒', 0.468762), ('黄药师', 0.461481)]
Related words of 葵花宝典	[('宝典', 0.580406), ('经书', 0.555128), ('九阴真经', 0.478705), ('琴谱', 0.448358), ('混上', 0.435583)]

表 2 Skip-gram模型结果

Related words of 韦小宝	[('康熙', 0.811152), ('索额图', 0.726793), ('小桂子', 0.692785), ('皇上', 0.721069), ('吴应熊', 0.703629)]
Related words of 郭靖	[('黄蓉', 0.872625), ('黄药师', 0.761286), ('杨过', 0.753109), (欧阳锋', 0.720267), ('黄蓉道', 0.686019)]
Related words of 杨过	[('小龙女', 0.904731), ('李莫愁', 0.786447), ('法王', 0.766600), ('黄蓉', 0.765817), ('金轮法王', 0.761639)]
Related words of 张无忌	[('赵敏', 0.866152), ('周芷若', 0.857216), ('小昭', 0.708666), ('赵姑娘', 0.694614), ('谢逊', 0.694141)]
Related words of 令狐冲	[(盈盈', 0.813138), ('岳不群', 0.801856), ('岳灵珊', 0.751617), ('任行', 0.736042), ('岳夫人', 0.722526)]
Related words of 峨嵋派	[('峨嵋', 0.748717), ('灭绝师太', 0.732069), ('武当派', 0.679899), ('本派', 0.651791), ('周芷若', 0.626593)]
Related words of 屠龙刀	[('倚天剑', 0.750354), ('谢逊', 0.741784), ('宝刀', 0.717537), ('屠龙', 0.677186), ('金花婆婆', 0.632721)]
Related words of 蛤蟆功	[('欧阳锋', 0.671509), ('西毒', 0.653830), ('九阴真经', 0.617519), ('一阳指', 0.558012), ('降龙十八掌', 0.546610)]
Related words of 葵花宝典	[('宝典', 0.669189), ('东方不败', 0.567427), ('秘笈', 0.562783), ('至高无上', 0.556782), ('日月神教', 0.541820)]

从人物关系上来看，Skip-gram的结果优于CBOW模型。例如，CBOW模型中，与“杨过”最相关的人物为“黄蓉”，然而两个人并非出现在同一部小说中，这可能是因为“黄蓉”也为主人公，且在训练语料库中多次出现，因此会被判定为与“杨过”相关性较强的人物。而Skip-gram模型结果中，人名最相关的词多与该人具有紧密联系。例如，“郭靖”最相关的词“黄蓉”为其爱人；“杨过”最相关的词为“小龙女”，两人为师徒关系；“令狐冲”最相关的词“盈盈”为其爱人岳盈盈的名等。

可以看出，CBOW模型，更注重文章整体的关联，很多相似度高的词跨越了多个时代、多本小说，而skip-gram模型更加注重局部，相似度高的词基本集中在同一本小说中。

2） 语义距离

分别计算“杨过，小龙女”，“杨过，东方”，“华山派，弟子”，“华山派，剑法”，“刀光，剑影”和“人民，剑影”之间的语义距离，实验结果如下：

表 3 CBOW结果

词向量对	语义距离
杨过，小龙女	0.82619816
杨过，东方	-0.03668024
华山派，弟子	0.39367643
华山派，剑法	0.31922108
刀光，剑影	0.3307692
人民，剑影	0.0013907112

表 4 Skip-gram结果

词向量对	语义距离
杨过，小龙女	0.9047312
杨过，东方	0.054001246
华山派，弟子	0.6380919
华山派，剑法	0.6392321
刀光，剑影	0.4258556
人民，剑影	0.033704497

CBOW模型和Skip-gram模型在判断输入词汇的语义距离时都具有较好的性能，但Skip-gram模型的相似度普遍更高，这是因为Skip-gram模型更加注重局部，相似度高的词基本集中在同一本小说中。

3） 相关词向量的 K-means 聚类

CBOW模型和Skip-gram模型得到的相关词向量的K-means聚类散点图如图所示。

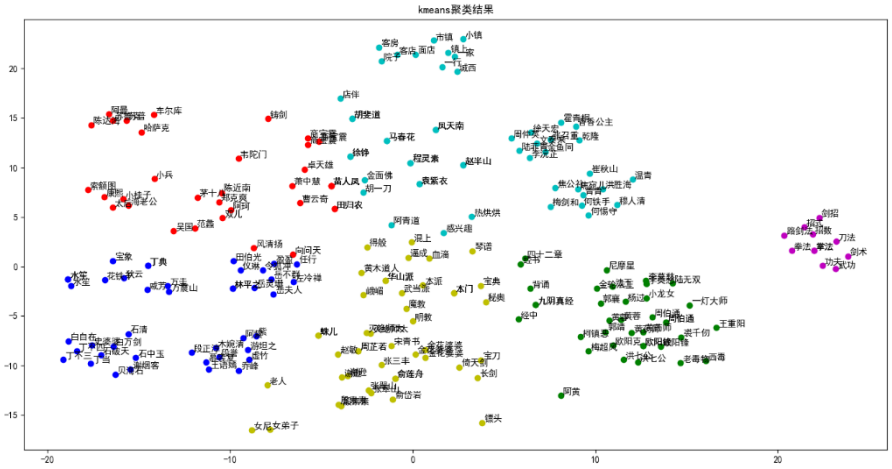


图 4 CBOW结果

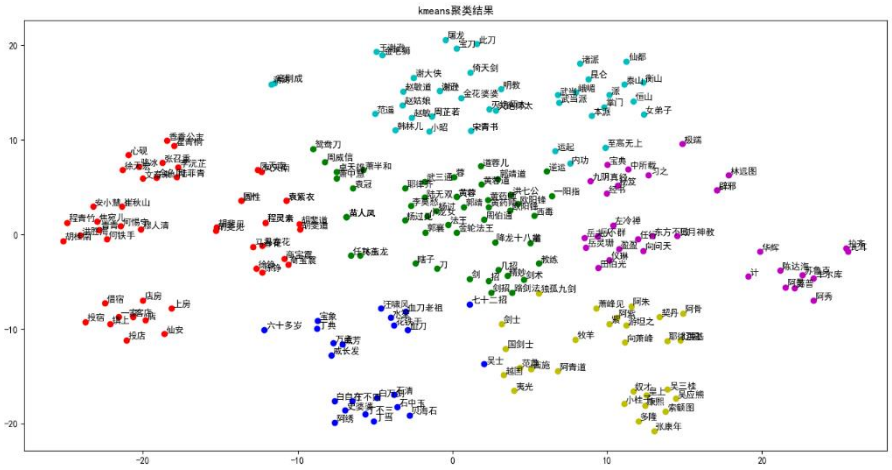


图 5 Skip-gram结果

考虑到显示效果因素，图中设定的n为6。从中可以清晰地看到同一本小说的词汇基本上被分在了一类，极少数有被分类错误的情况，由此可见，整体效果良好，且充分验证了词向量的有效性。

4） 某些段落直接的语意关联

分别利用语料库中段落以及其他语料段落计算语义关联，语料库段落为：

'黄蓉不答，拉著他手走到後院，四下瞧了瞧，这才说道：“你和过儿的对答，我在窗外都听见啦。他不怀好意，你知道麽？”郭靖吃了一惊，问道：“甚麽不怀好意？”黄蓉道：“我听他言中之意，早在疑心咱俩害死了他爹爹。”郭靖道：“他或许确有疑心，但我已答允将他父亲逝世的情由详细说给他知道。”黄蓉道：“你当真要毫不隐瞒的说给他听？”郭靖道：“他父亲死得这麽惨，我心中一直自责。杨康兄弟虽然误入歧途，但咱们也没好好劝他，没想法子挽救。”黄蓉哼了一声，道：“这样的人又有甚麽可救的？我只恨杀他不早，否

则你那几位师父又何致命丧桃花岛上？”郭靖想到这桩恨事，不禁长长叹了口气。'

'黄蓉道：“朱大哥叫芙儿来跟我说，这次过儿来到襄阳，神气中很透著点儿古怪，又说你和他同榻而眠。我担心有何意外，一直守在你窗下。我瞧还是别跟他睡在一房的好，须知人心难测，而他父亲……总是因为一掌拍在我肩头，这才中毒而死。”郭靖道：“那可不能说是你害死他的啊。”黄蓉道：“既然你我均有杀他之心，结果他也因我而死，那麽是否咱们亲自下手，也没多大分别。”郭靖沉思半晌，道：“你说得对。那麽我还是不跟他明言的为是。蓉儿，你累了半夜，快回房休息罢。过了今晚，明日我搬到军营中睡。”'

其他语料段落为：

'中国共产党已走过百年奋斗历程。我们党立志于中华民族千秋伟业，致力于人类和平与发展崇高事业，责任无比重大，使命无上光荣。全党同志务必不忘初心、牢记使命，务必谦虚谨慎、艰苦奋斗，务必敢于斗争、善于斗争，坚定历史自信，增强历史主动，谱写新时代中国特色社会主义更加绚丽的华章。'

'十九大以来的五年，是极不寻常、极不平凡的五年。党中央统筹中华民族伟大复兴战略全局和世界百年未有之大变局，就党和国家事业发展作出重大战略部署，团结带领全党全军全国各族人民有效应对严峻复杂的国际形势和接踵而至的巨大风险挑战，以奋发有为的精神把新时代中国特色社会主义不断推向前进。'

计算关联度结果为：

表 5 CBOW结果

段落对	语义距离
语料库随机抽取	0.82619816
其他语料库	-0.03668024

可以看到，CBOW只有对语料库内抽取的段落关联性较强。

Conclusions

1. CBOW 和 Skip-gram 均能够实现相关词汇聚类的功能，但人物关系上来看，Skip-gram 的结果优于 CBOW 模型。例如，CBOW 模型中，与“杨过”最相关的人物为“黄蓉”，然而两个人并非出现在同一部小说中，这可能是因为“黄蓉”也为主人公，且在训练语料库中多次出现，因此会被判定为与“杨过”相关性较强的人物。而 Skip-gram 模型结果中，人名最相关的词多与该人具有紧密联系。例如，“郭靖”最相关的词“黄蓉”为其爱人；“杨过”最相关的词为“小龙女”，两人为师徒关系；“令狐冲”最相关的词“盈盈”为其爱人岳盈盈的名等。可以看出，CBOW 模型，更注重文章整体的关联，很多相似度高的词跨越了多个时代、多本小说，而 skip-gram 模型更加注重局部，相似度高的词基本集中在同一本小说中。
2. CBOW 和 Skip-gram 在判断输入词汇的语义距离时都具有较好的性能，但 Skip-gram 模型的相似度普遍更高，这是因为 Skip-gram 模型更加注重局部，相似度高的词基本集中在同一本小说中。
3. CBOW 只有对语料库内抽取的段落关联性较强。
4. Word2Vec 模型生成的词向量能够正确表示词语与词语之间的语义关系，生成合适的词向量。

References

[1] https://blog.csdn.net/v_JULY_v/article/details/102708459
[2] https://blog.csdn.net/apr15/article/details/107991954
[3] https://cloud.tencent.com/developer/article/2086601
[4] https://blog.csdn.net/yangbindxj/article/details/123911869
[5] https://easyai.tech/ai-definition/word2vec/