## 1，对所写的代码的解释

(1),

Word2vector.py代码主要是用来对词向量模型进行初始化、训练和评估，以下是各个部分的简要说明：

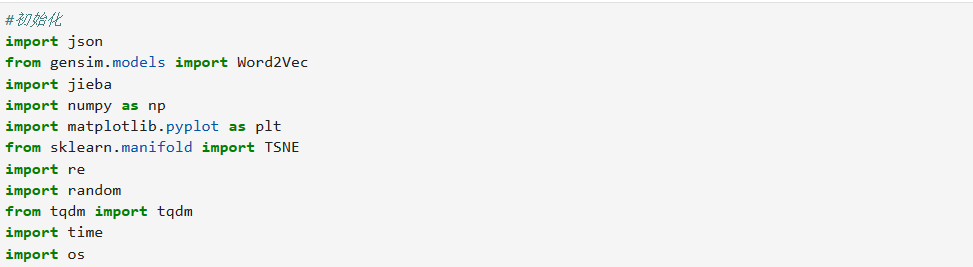
* 初始化部分：导入所需的库，并定义了一系列函数和类来实现词向量模型的训练、评估和可视化。
* display函数：用于展示数据。
* read\_json函数：用于从JSON文件中读取数据。
* select\_model函数：根据模型名称选择对应的模型，如Word2Vec或FastText。

Model类：包含了模型的初始化、预训练、训练和评估方法。

* \_\_init\_\_方法：初始化模型参数。
* cut\_words方法：对文本数据进行分词处理。
* pre\_train\_model方法：预训练一个词向量模型。
* random\_samples\_trainmodel方法：使用随机采样进行模型训练。
* epoches\_samples\_trainmodel方法：将数据分批进行模型训练。
* prove\_model方法：评估模型性能。

versionism类：包含了对训练好的模型进行可视化和评估的方法。

* \_\_init\_\_方法：初始化模型。
* demosion\_h22方法：使用t-SNE对词向量进行降维并保存。
* vector\_2d\_show方法：展示词向量的二维可视化结果。
* cpka方法：展示两个词之间的余弦相似度，并在二维可视化图上标记这两个词及其相似词。

















(2),

Kmeans.py代码是从预训练的Word2Vec模型中加载词向量，并使用K均值算法对词向量进行聚类。接下来，它找到每个聚类中心最近的词汇，并可视化这些词汇。

首先，它加载了一个预训练的Word2Vec模型，然后获取模型中所有词语的词向量，并将其存储在vectors变量中。

接下来，它设置了聚类的数量（num\_clusters），并使用K均值算法对词向量进行聚类。聚类后，每个词向量将被分配一个聚类标签。

然后，它将每个词语与其聚类标签和词向量一起存储在word\_cluster\_map列表中。

接下来，它分别提取了两个聚类的词语，并在二维空间中进行可视化。每个聚类的词语用不同的颜色表示，以便在图中进行区分。

最后，它计算了每个聚类中心最近的词汇，并在二维空间中进行了可视化。对于每个聚类中心，它找到了与之距离最近的词语，并在图中标记出来。同时，它还标注了这些最近词汇的标签。







## 2，实验过程和心得体会

**A，训练数据的预处理**

首先读取数据：



观察数据，可知爬取的网络文字数据以字典的方式存储。字典中有五个key内容：qid，category，title，desc，answer。考虑到句子的完整性和段落的可理解性，answer内容部分为最优选择。同时考虑到句子中的标点符号之类的数据对分词结果作用不大，于是去掉标点符号，并进行结巴分词。示例结果如下：



**B,模型训练的阶段：**

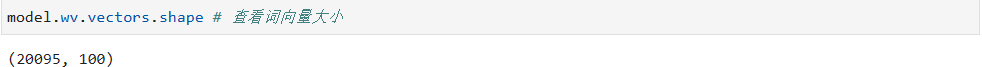
B1：截取一段数据进行训练





结果如下：

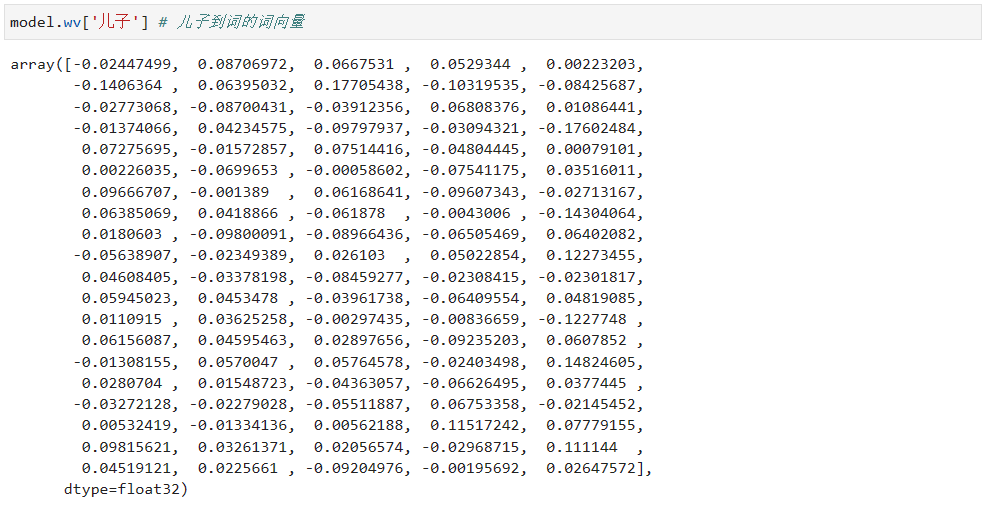
（1）词向量大小：



（2）获得的所有词汇组：



（3）词向量的形式

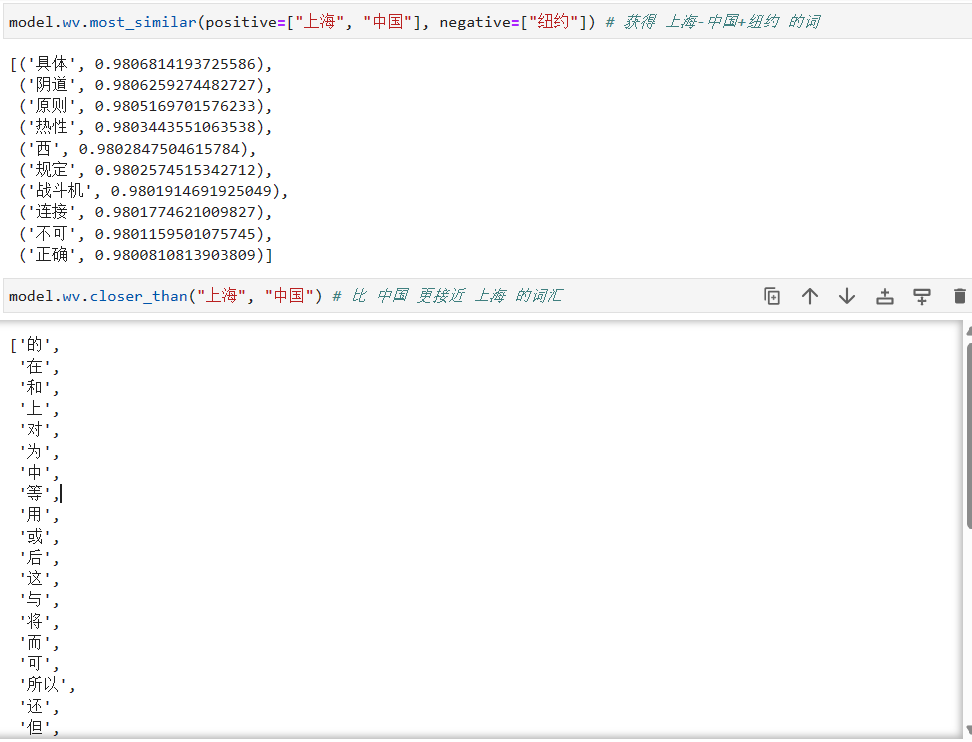


（4）词向量相似度分析





（5）词向量的向量加减与比较



**B2,随机抽取若干数据进行N轮训练**

模型训练伪代码：

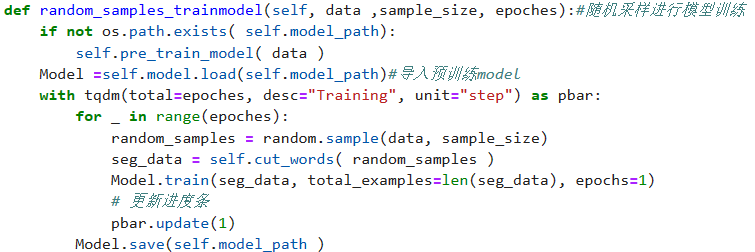
For t :0-N#对于N轮训练

Train\_data = random（sample\_size）#随机抽取size大小数据集

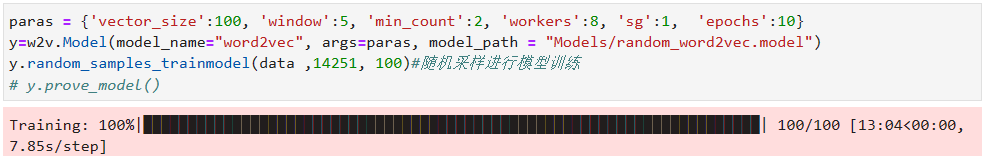
Model.train(train\_data)#利用数据集进行一轮训练

Model.save#模型储存

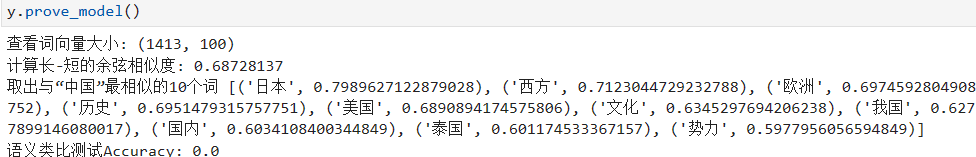
代码如下：



训练采用skip-gram算法：



结果如下：



以及：



可以知道随机采样训练会不可比避免的漏掉某些词，以及在词向量相似度，语义类比测试上表现并不理想。

**B3,分批次对所有数据进行N轮训练**

模型训练伪代码：

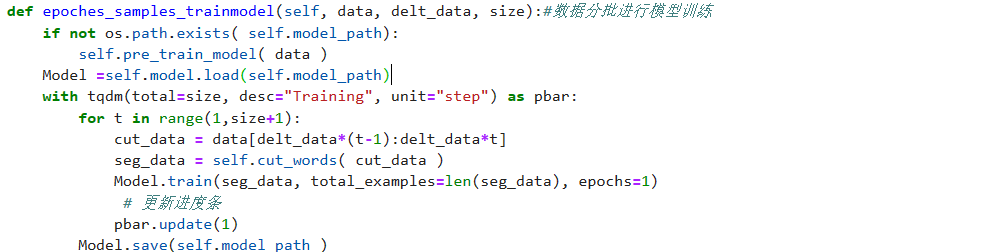
For t :0-N#对于N轮训练

Train\_data = data[ t\*size : (t+1)\*size ]#依次截取总数据data中size大小数据集

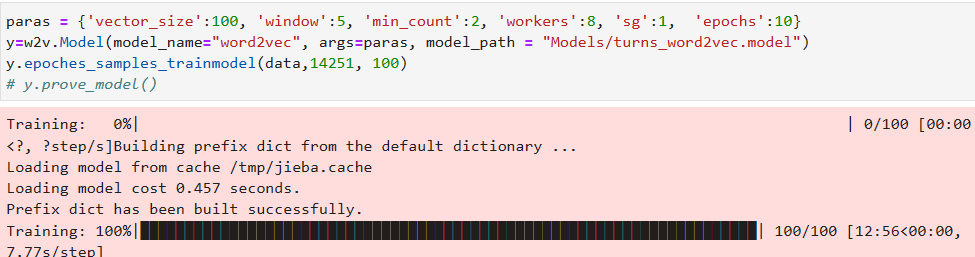
Model.train(train\_data)#利用数据集进行一轮训练

Model.save#模型储存

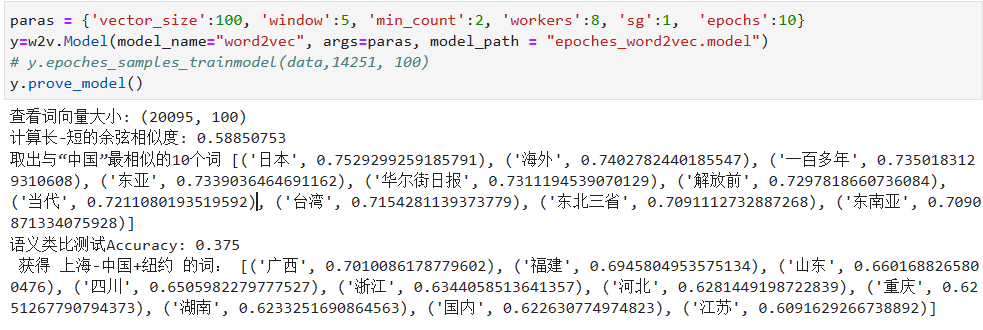
代码如下：



训练采用skip-gram算法：



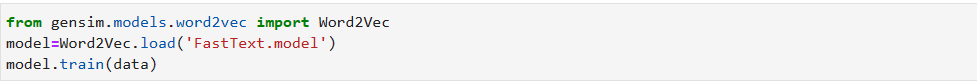
结果如下：



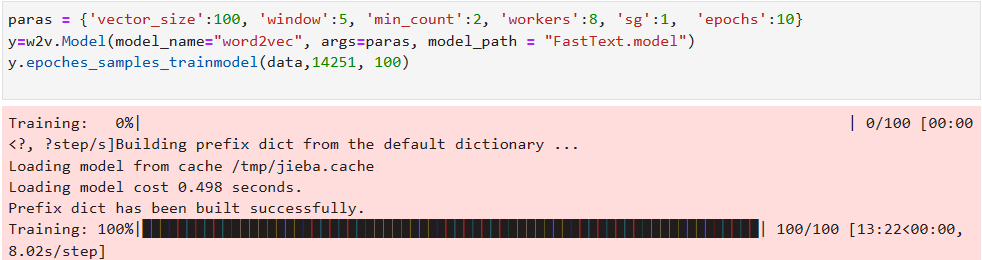
可以看到余弦相似度下降，语义类比测试准确率上升，“纽约”词向量也存在，说明分批次的训练优于随机抽取训练。

**B4,采用预训练模型进行分批次训练**

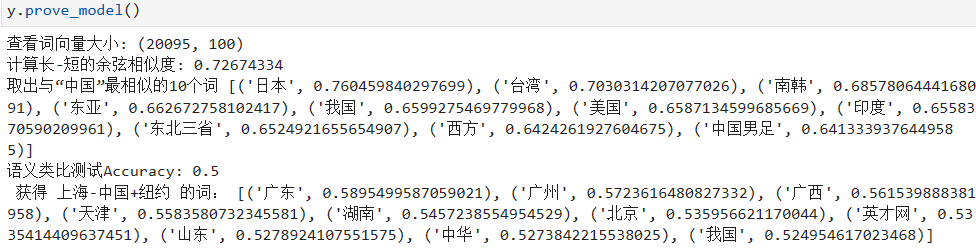
为了进一步的改善训练的模型性能，增加预训练的模型：



其中FastText.model是预训练的model。下面利用Model模块进行正式的训练：



结果如下：



可以看到语义类比测试准确率上升，词向量的相似度也更合理。

**C,对实验参数的调整**

该实验部分将会详细在选做部分《分析 Word2Vec 中不同超参数对最后分词效果的影响》详细分析，请下翻到该部分。

**D,实验心得：**

数据预处理的重要性：在进行Word2Vec实验之前，数据预处理是至关重要的一步。包括分词、去除停用词、处理低频词等操作，可以帮助模型更好地捕捉单词之间的语义信息。

参数选择对结果影响显著：Word2Vec有一些关键参数，如向量维度、窗口大小、负采样个数等。不同的参数选择可能导致不同的结果。在实验过程中，需要进行一些参数调优的工作，以获得最佳的模型效果。

模型训练时间：Word2Vec模型的训练可能需要一定的时间，特别是在大规模数据集上。因此，在进行实验时需要考虑到训练时间，以及合理地安排实验的时间计划。

词向量的可视化：通过降维和可视化技术，可以将高维的词向量投影到二维或三维空间中，以便更直观地观察词向量之间的关系。这有助于理解模型学到的词义和语义关系。

模型评估与应用：在实验完成后，需要对模型进行评估，以确定其性能和效果。常用的评估指标包括词类比、词相似度等。另外，还可以将训练好的词向量应用到具体的任务中，如文本分类、信息检索等。

调整训练目标：Word2Vec有两种训练目标，分别是Skip-gram和CBOW。它们各自适用于不同的场景，需要根据具体任务选择合适的模型结构和训练目标。

持续学习与优化：Word2Vec是一个成熟的技术，但仍然有很多改进的空间。在实验过程中，可以尝试一些新的方法和技巧，不断优化模型，提升词向量的质量和表征能力。

总的来说，Word2Vec是一个非常有用的工具，可以帮助我们从文本数据中提取出丰富的语义信息。通过实验和不断的探索，我们可以更好地理解和应用这一技术，从而提升文本处理和自然语言理解的能力。

## 3，对实验结果的分析

长-短语义相似度：范围为[0,1]，最相似时值为1

相似词合理程度：根据判断者常识，对相似词结果的合理程度打低，中，高三个评分。

语义类比测试准确率：范围为[0,1]，最准确时值为1

词向量加减得到新词的合理程度：根据判断者常识，对相似词结果的合理程度打低，中，高三个评分。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 训练方式\评测标准 | ‘长-短’语义相似度 | 相似词合理程度 | 语义类比测试准确率 | 词向量加减得到新词的合理程度 |
| 截取一段数据训练 | 0.99 | 低 | 0 | 低 |
| 随机抽取若干数据进行N轮训练 | 0.69 | 中 | 0 | 无结果 |
| 分批次对所有数据进行N轮训练 | 0.59 | 高 | 0.375 | 高 |
| 采用预训练模型进行分批次训练 | 0.73 | 高 | 0.5 | 高 |

‘长-短’语义相似度在0.7左右可以认为预测合理，所以可以知道：随机抽取若干数据进行N轮训练；采用预训练模型进行分批次训练两种方式预测最好；分批次对所有数据进行N轮训练稍差；截取一段数据训练方式最不合理。

相似词合理程度的评价标准下，分批次对所有数据进行N轮训练，采用预训练模型进行分批次训练两种方式是最好的；随机抽取若干数据进行N轮训练方式稍差；截取一段数据训练方式最差。

语义类比测试准确率：截取一段数据训练，随机抽取若干数据进行N轮训练，分批次对所有数据进行N轮训练，采用预训练模型进行分批次训练，四种方式准确率依次上升。

词向量加减得到新词的合理程度：截取一段数据训练方式得到结果不合理；随机抽取若干数据进行N轮训练的方式因为没有‘纽约’词向量，所以没有得到结果；分批次对所有数据进行N轮训练，采用预训练模型进行分批次训练，两种方式得到结果很合理。

综合来看，采用预训练模型进行分批次训练的方式在所有评测标准下表现最好，其次是分批次对所有数据进行N轮训练，再次是随机抽取若干数据进行N轮训练的方式。截取一段数据训练的方式表现最差。

## 4，对课堂所学知识的理解

**1，CBOW和Skip-gram两种训练模型：**

CBOW模型通过上下文预测目标词汇，而Skip-gram模型则是通过目标词汇预测上下文。

在CBOW模型中，输入是上下文中的词向量的平均值，输出是目标词汇的词向量。而在Skip-gram模型中，输入是目标词汇的词向量，输出是上下文中词语的词向量。

CBOW模型的优点是对常见词汇的学习效果好，训练速度快，尤其适用于小规模数据集。

相比于CBOW，Skip-gram模型更适合于大规模数据集，尤其是对于罕见词汇的学习效果更好。

**2，上下文窗口：**

在Word2Vec的训练中，需要定义一个上下文窗口大小。这个窗口大小决定了模型在预测目标词时会考虑多远的上下文词汇。

窗口大小的选择直接影响了模型对于上下文的理解程度：较大的上下文窗口可以捕捉更丰富的语义信息，但也可能导致模型学习到更多的噪声信息；而较小的上下文窗口则可能导致词向量的语义表示不够准确。

大的上下文窗口可以捕捉更广泛的语境信息，但也可能增加训练的计算复杂度；而较小的上下文窗口则可能导致模型忽略了一些重要的语境信息。

**3，负采样：**

Word2Vec中的负采样技术用于加速训练过程。负采样通过减少训练样本数量，从而降低了计算成本。在负采样中，对于每个正样本，会随机选择一些负样本，用来训练模型。

正样本是从训练数据中真实的（或者说是观察到的）词汇对，而负样本则是根据一定的概率分布进行采样的虚拟的词汇对。

负采样中每个词汇被选为负样本的概率与其频率有关，通常频率越高的词汇被选为负样本的概率越低。这样做的目的是让低频词汇有更多的机会被选择为负样本，以提高对低频词汇的建模效果。

**4，向量空间的表示：**

Word2Vec通过训练得到的词向量，将单词嵌入到一个高维向量空间中。在这个向量空间中，词汇的相似性可以通过向量之间的距离来衡量，比如常用的余弦相似度。利用向量空间，Word2Vec模型有以下功能：

1）语义关系：

Word2Vec生成的词向量能够捕捉到单词之间的语义关系，例如语义上相似的单词在向量空间中会更加接近，而语义上不相关的单词则会远离。这种语义关系的表示可以用于各种自然语言处理任务中，如文本分类、信息检索等。

2）情感分析：

在情感分析任务中，Word2Vec可以用于学习词语之间的情感关系。通过训练得到的词向量，可以将文本中的情感词嵌入到向量空间中，从而进行情感分类或情感极性分析。

3）关系抽取：

Word2Vec可以用于关系抽取任务，通过学习词语之间的语义关系，可以帮助模型更好地理解文本中的关系信息，从而提取出其中隐藏的关系。

4）文本生成与推荐：

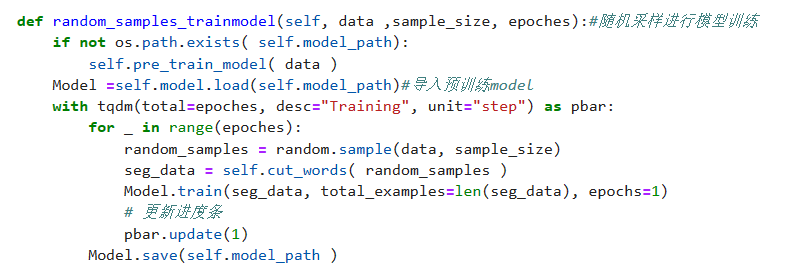
Word2Vec生成的词向量可以用于生成文本和推荐系统。在文本生成任务中，可以利用词向量来生成语义合理的文本。在推荐系统中，可以通过分析用户和文本的词向量，提高推荐的精度和个性化程度。

## 5，选做

**1，分析降低训练代价，提升训练性能的方式**

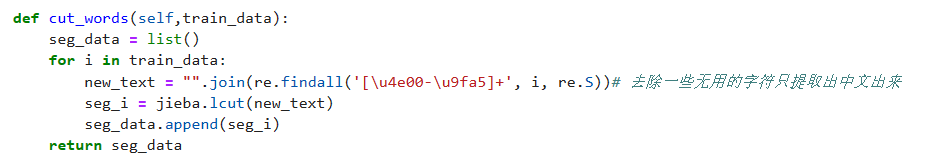
A，减小数据规模：

如果训练数据量非常大，考虑对数据进行采样或者截断，以降低数据量。例如，可以只使用部分语料库进行训练，或者通过简单的随机采样来减少数据量。



B，数据预处理：

在输入数据到模型之前，进行一些预处理操作，如词语的小写化、去除停用词、词干提取或者词语标准化等。这可以减少词汇量，并且提高模型的泛化能力。或者进行分块处理，将语料库分成多个块，每次加载一个块进行训练，这样可以减少内存消耗，并且使得可以处理更大规模的数据。



C，调整模型参数：

通过调整词向量的维度、上下文窗口大小、负采样的数量等参数，可以探索不同的模型设置，从而提高性能。

D，使用更高效的实现：

尝试使用更高效的实现来加速训练过程。例如，使用 Gensim 中的多线程实现或者 Cython 加速版本。这些实现通常比基本的 Python 实现更快速。或者来块黄老板的AI芯片，可惜钱袋空空。

E，使用更小的词向量维度：

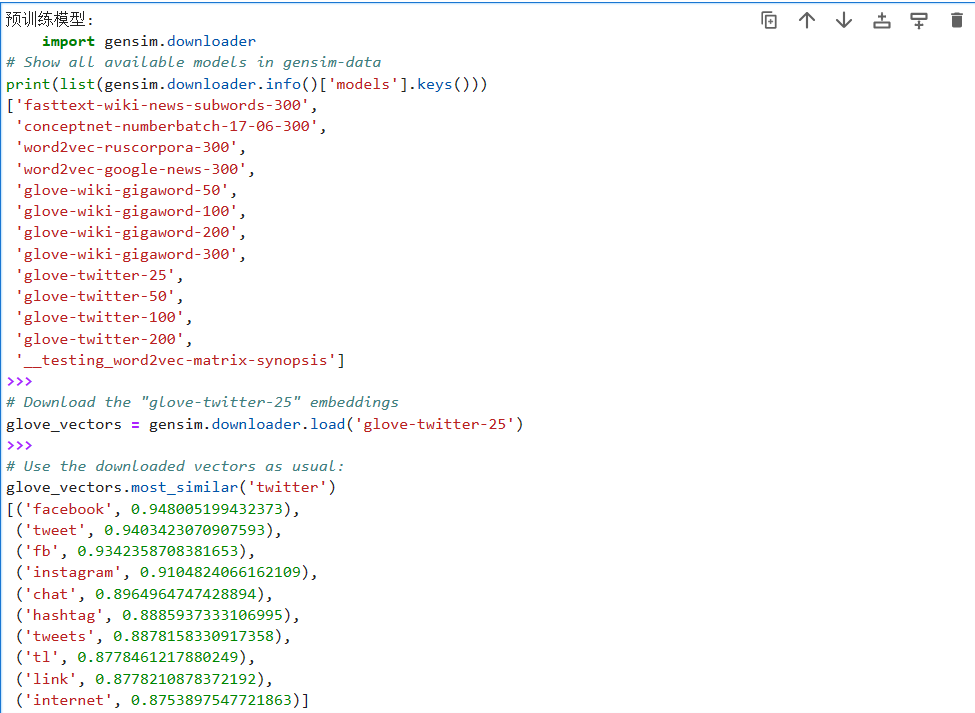
减少词向量的维度可以减少模型的复杂度，从而降低训练代价。虽然较小的词向量维度可能会损失一些信息，但对于大规模语料库来说，这种损失可能是可以接受的。

F，迭代训练：

在大规模语料库上进行单次完整的训练可能是昂贵的。相反，可以选择使用迭代训练的方式，即多次遍历数据集进行训练，每次只使用部分数据。这样可以减少单次训练的数据量，降低训练代价。

G，使用预训练模型：

如果应用场景允许，可以考虑使用预训练的 Word2Vec 模型，而不是从头开始训练。这样可以节省训练时间，并且通常能够获得更好的词向量表示，尤其是在资源有限的情况下。



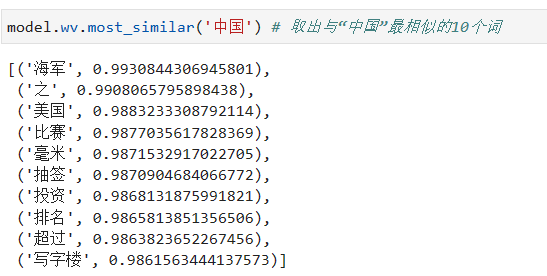
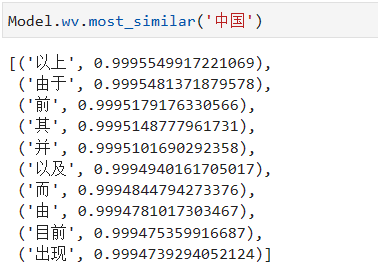
**2，分析 Word2Vec 中不同超参数对最后分词效果的影响**

1)，epochs 参数被设置为 10，这意味着模型将在整个语料库上进行 10次迭代训练。



截取数据的方式进行训练模型：

Epochs == 10 Epochs == 1



可以看出：在训练数据较小的情况下，Epochs == 10的结果更合理。

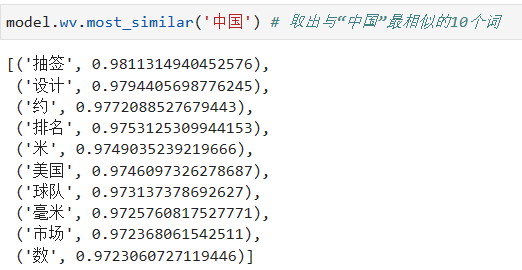
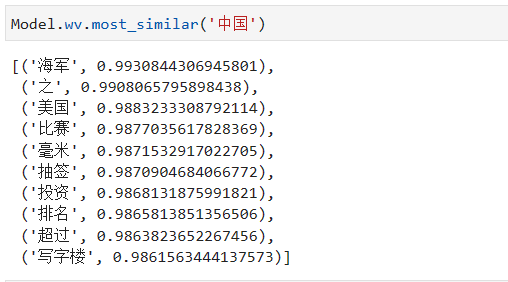
对epochs的理解：

|  |  |
| --- | --- |
| 收敛速度： | 较少的迭代次数可能会导致模型收敛速度过慢，而较多的迭代次数可以加速模型的收敛。通常情况下，增加迭代次数可以使模型更快地学习到数据中的模式和规律，从而提高训练效率。 |
| 泛化能力： | 适当的迭代次数可以帮助模型更好地泛化到未见过的数据，从而提高模型的泛化能力。较少的迭代次数可能会导致模型欠拟合，而较多的迭代次数可以减少欠拟合的风险，提高模型的泛化性能。 |
| 词向量质量： | 较多的迭代次数通常可以得到更好质量的词向量表示，因为模型有更多的机会学习到数据中的信息。增加迭代次数可以使词向量更加准确和丰富，从而提高模型在各种语言任务中的性能。 |
| 过拟合风险： | 过多的迭代次数可能会导致模型过拟合，即在训练数据上表现良好，但在测试数据上表现较差。 |

2)，较大的窗口大小可以捕获更广泛的语义信息，但可能会导致模型过度拟合。较小的窗口大小可能会限制模型的上下文理解能力。



window=10 and epochs=10 window=5 and epochs=10

window=10后，[抽签，排名，美国]等词语信息依然还在，但是其他词汇有变化。可以理解为当窗口增大后，在保留附近基本词义的情况下，在广泛的上下文找到了新的相似词。

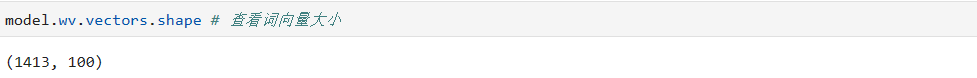
3)，较高的维度可以提供更丰富的语义信息，但也会增加模型的复杂度和训练时间。较低的维度可能会限制模型的表达能力。Word2Vec参数中向量维度vector\_size=100，在分批次训练100次的方式下训练，时间已经很长。于是便不进行增加向量维度的实验。

4)，workers=8，设置电脑cpu的8个内核并行计算。

5)，min\_count=10，min\_count 参数用于控制过滤词频的阈值。具体来说，它指定了在训练过程中忽略那些词频少于 min\_count 的词语。这个参数的作用在于过滤掉出现次数过少的词语，从而减少模型的噪声，提高模型的效率和性能。



当设置min\_count==10时进行训练，得到的结果有：



词向量数量显著减少，可知训练数据量不算大。

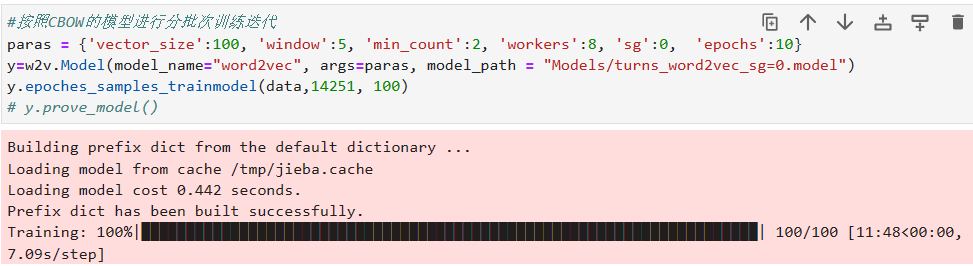


‘纽约’词向量在词汇里面找不到，可以知道‘纽约’一词出现的频率很小，少于10次，被过滤。由于训练数据少，不需要设太大的min\_count来过滤词汇，笔者认为min\_count==2已经很好。

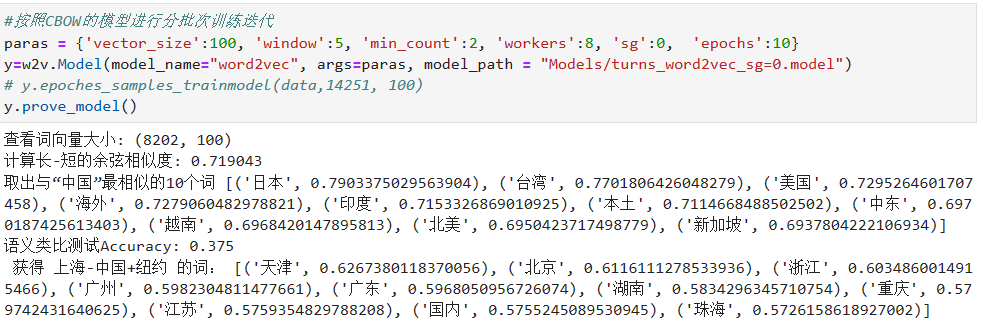
7)，max\_vocab\_size 参数来限制词汇表的大小，从而间接过滤掉一些高频词汇。这个参数指定了词汇表中最多允许包含的词语数量。但由于不好确定最大词汇数量阈值，所以没有设置。

**3，分析CBOW 和 skip-gram两种不同实现方式的优劣**

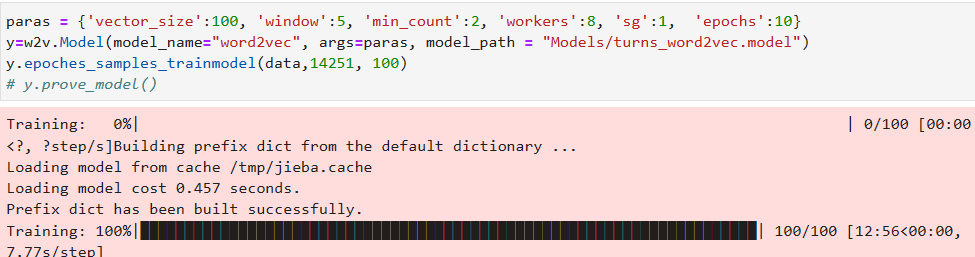
**CBOW 模型**的工作原理是基于上下文中的词语来预测目标词汇。具体来说，它假设在给定上下文词语的情况下，预测目标词汇。CBOW 模型的架构包含一个隐藏层，其维度通常是词向量维度的一半，用于学习词语之间的语义关系。CBOW 模型通过最大化预测目标词汇的概率来训练模型参数。设置Word2vector模型训练参数,模型会以CBOW的形式训练。以下是训练过程和结果：



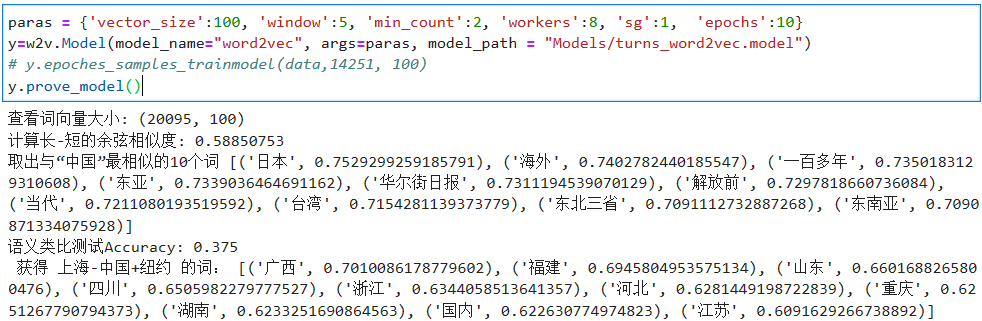
用时11分48秒。



**Skip-gram 模型**的工作原理与 CBOW 相反，它的目标是根据目标词汇来预测上下文。具体来说，给定一个目标词汇，skip-gram 模型试图预测它的上下文词语。Skip-gram 模型同样包含一个隐藏层，用于学习词语之间的语义关系。设置Word2vector模型训练参数,模型会以skip-gram的形式训练。以下是训练过程和结果：



用时12分56秒。



对比两种模型训练结果差距：

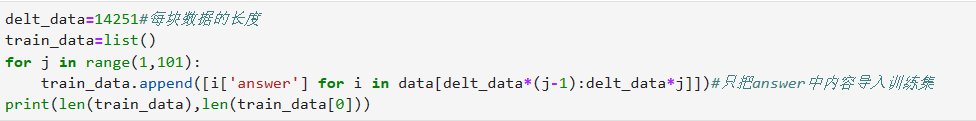
在CBOW模型的结果中，词向量大小相比于skip-gram训练的词向量大小，少于其一半，也就是说减少很多；长-短余弦相似度，根据上面结果分析部分可知0.7左右为佳，说明CBOW模型训练的结果效能有上升；在语义类别准确率和向量加减分析中表现差不多；在训练时间上，CBOW比Skip-gram早结束一分钟，可以认为CBOW的训练速度更快。

查阅资料后分析优劣可知：

|  |  |  |
| --- | --- | --- |
|  | 优势 | 劣势 |
| Skip-gram： | 考虑词序信息：Skip-gram 模型考虑了词语的顺序信息，因此在处理长文本时通常比 CBOW 模型更有效。  语义信息丰富：由于 Skip-gram 模型是基于目标词汇来预测上下文，因此它可以更好地捕捉词语之间的语义关系。 | 训练速度较慢：Skip-gram 模型通常需要更多的训练时间，因为它需要为每个目标词汇预测其上下文。  对稀有词的处理较差：由于 Skip-gram 模型在训练过程中需要考虑每个目标词汇的上下文，因此对于稀有词的处理可能不如 CBOW 模型有效。 |
| CBOW ： | 训练速度快：由于 CBOW 模型的目标是根据上下文预测目标词汇，因此它通常比 skip-gram 模型训练速度更快。  对稀有词的处理：CBOW 模型对稀有词的处理较为有效，因为它在训练过程中将上下文的信息汇总到一个目标词汇上，从而减轻了稀有词的影响。 | 语义信息损失：由于 CBOW 模型将上下文信息汇总到一个目标词汇上，因此它可能会丢失一些词语之间的语义关系。  不适合长文本：CBOW 模型在处理长文本时可能效果不佳，因为它没有考虑到词语的顺序。 |

**4，如何更高效地利用数据**

A，分块处理： 语料库非常庞大，可能无法一次性加载到内存中进行训练。在这种情况下，将语料库分成多个块，每次加载一个块进行训练。 这样可以减少内存消耗，并且使得可以处理更大规模的数据。



B，词频过滤： 可以根据词频对语料库进行过滤，删除出现频率太低或太高的词语。出现频率太低的词语可能是噪音，对模型训练没有太大帮助，而出现频率太高的词语可能是停用词，对模型训练也没有太大意义。通过删除这些词语，可以减少数据量并提高模型的效率： 利用new\_text = "".join(re.findall('[\u4e00-\u9fa5]+', text3, re.S))去除一些无用的字符只提取出中文出来。

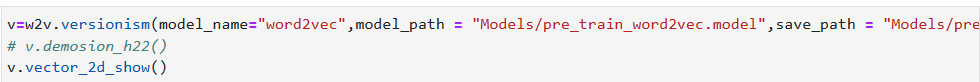


C, 随机采样：在处理大型语料库时，通常可以使用随机采样来减少数据量。设置一个采样率，例如保留原始数据的百分之几，然后随机选择数据进行训练。这样可以减少数据量，从而加快训练速度，并且在某些情况下，可以提高模型的泛化能力。

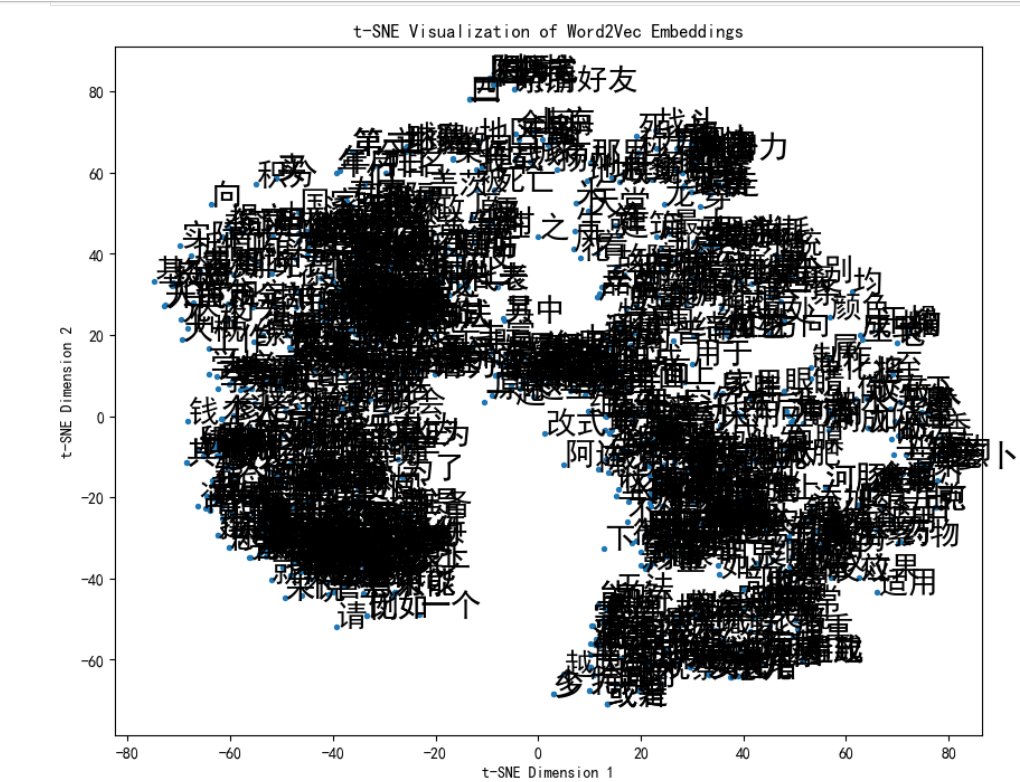


**5，结果可视化分析**

A，首先进行向量降维，然后进行可视化展示。



B，可视化结果



**6，其他感兴趣的实验**

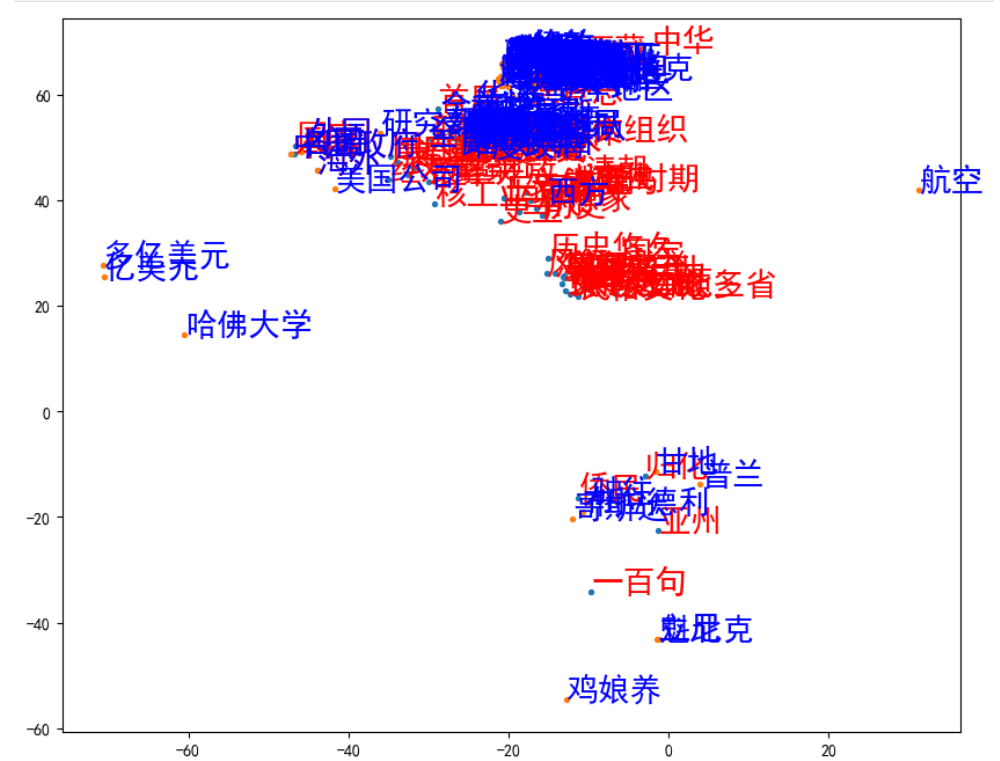
（1）“中国”与“美国“相似度的可视化展示



结果有：

蓝色-美国

红色-中国

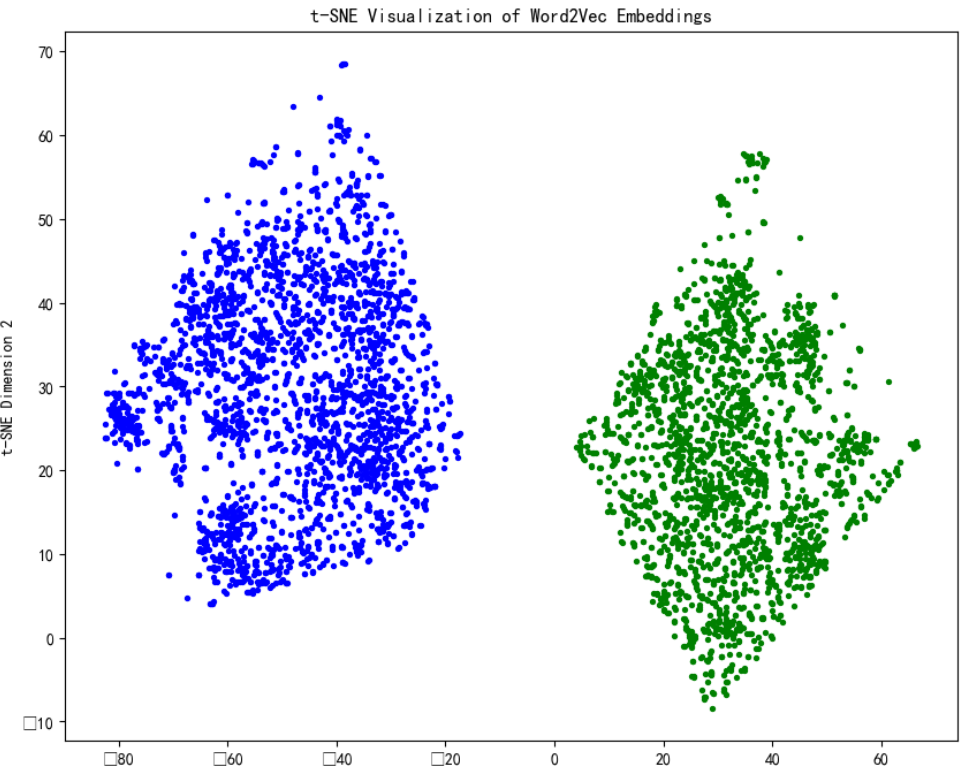




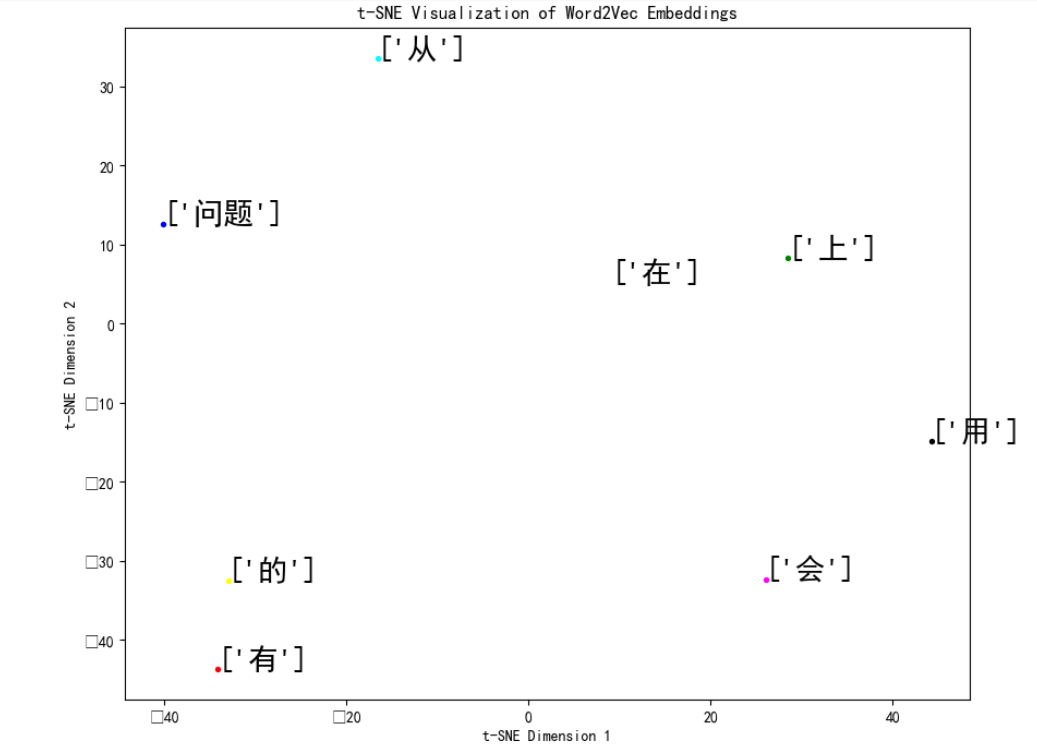
可以看到中国-美国的相识范围有很大重合。

（2）K-means聚类：

聚类后，对其中两类数据可视化：



之后，取出所有离簇中心位置最近的词，并可视化+贴上标签：



形容词：上

介词：在，的，从

名词:问题

动词：用，有，会

猜测：语句的结构主要是动词+介词构成，形容词和名词多在句子中呈修补作用。