电子科技大学 计算机 学院

标 准 实 验 报 告

（实验）课程名称 人工智能综合实验 I

学生姓名：刘洋岑　　学 号： 2020080601018

指导教师：姬艳丽

实验地点：主楼 A2-412

实验时间：2022.3.24

电子科技大学教务处制表

电 子 科 技 大 学

实 验 报 告

一、实验项目名称：K 聚类实现分类任务

二、实验内容：

实验一：编程实现 K 聚类分类器模型，在西瓜数据集 3.0α（表 4.5）上验证 K 近邻分类器。

（1）实现 K 近邻分类器，在西瓜数据集进行测试。

（2）根据 WSS 参数进行最优 K 值的测试。

实验二（补充实验）：对于 NLP 中的词袋聚类问题进行简单的实验。

（“作者”： 刘洋岑（2020080601018） 谭一谈（2020080601022））

（1）实现中文文本数据集的分词（基于简化版 Node2vec 技术）

（2）使用 deep walking 实现词语关系建模

（3）实现所有词语的 one-hot 编码到 embedded 编码的映射，得到词袋向量。

（4）使用 K-means 对于词袋向量进行聚类，并可视化。

三、实验算法设计：

实验一

由于西瓜数据集本身比较小，所以仅仅使用了 KNN 直接进行套用。大致的算法为：

1.对随机选取 k 个样本作为初始的聚类中心

2.计算每个样本到各个聚类中心之间的距离，将每个样本分配给距离它最近的聚类中心，此时全部样本已划分为 k 组

3.更新聚类中心，将每组中样本的均值作为该组新的聚类中心；

4.重复进行第二、三步，直到聚类中心趋于稳定，或者到达最大迭代次数。

实验二（补充实验）

此部分与谭一谈（2020080601022 合作完成），其中刘洋岑完成思路构建、deep walking 和 node embedding 的深度学习部分，谭一谈完成分词、文本词语建图、K-means(based on cosine similarity)以及结果可视化部分。

针对于 NLP 中常见的词语拓扑关系建模以及分析问题，我与合作者（谭一谈 2020080601022）查阅了大量的相关资料，得到了一种基本的词聚类方法。

词聚类任务是从大量文本中得到相关性较强词语的分布关系，其下游任务包含：词语情感分析、词语逻辑联系、文本预测等。我们这里仅仅使用最基本的词袋提取方法：Node2vec，

对于找到的小规模文本进行词向量提取，最终的工作是使用这个方法对词向量进行了K-means 聚类。

1. 中文分词处理：

使用了 Python 中的 nltk 库进行文本词语的分词，并且对于每一个词语得到了唯一的 one-hot 编码。

2. Node2vec word embedding 方法：

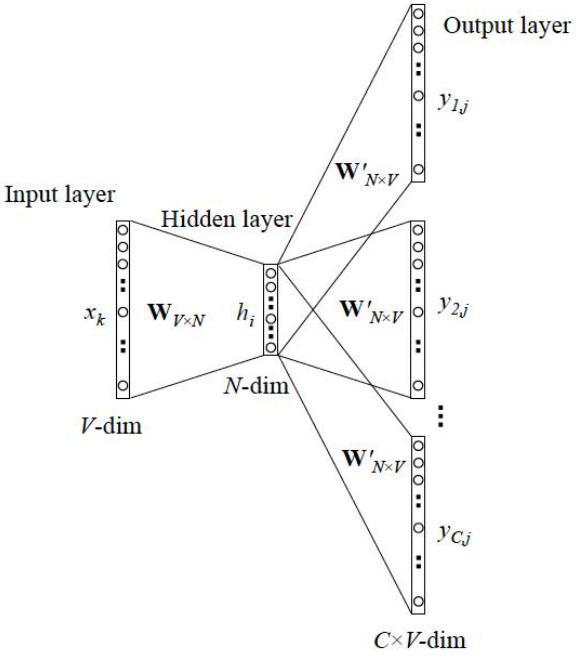
2.1. Deep walk 方法

对于词语的 one-hot 编码， 无法计算出两个词语之间的联系。我们使用 one-hot 编码进行映射得到低纬度的高信息密度的词袋编码。首先对于文本进行 deep walk。这个方法使用了一个固定长度的滑动窗口，在文本中进行相邻词语的记录，得到多条 path。我们对这个方法进行了更合理的改进（使用了 Word2vec 后来引申出的 Node2vec 方法）将文本中每个词语建模为一个 node，每两个在文本中相邻的词语（点）连无向边边（相邻的点间可能有多条边），最终得到一个完整的图。

在图中我们从每个点出发，每次进行固定次数的路径读取。每次路径的运行规则根据这个点的出度（以及指向自己的出度）进行均一化的概率的行走。最终每个点都记录了与各个其他点的初步联系（行走次数越多，结果越鲁棒）。最终将这个每个点的路径进行均一化（一般是用 SoftMax，我们仅仅使用了简单的除法）。

2.2. Word embedding 方法

根据每个 one-hot 到 path 进行一个双层线性映射，其中隐藏层的神经元不带激活函数（保证线性映射）隐藏层神经元数目为我们期望得到的 embedded vector 维度。（一共有 K 个词语，x 为 one-hot 编码，y 为得到的 path，h 是隐藏层）



图中 W 即为最后的词袋。其中的训练过程如下：

$$output_i = x_i WW'$$
$$loss_i = \|y_i - output_i\|^2$$
$$loss = \sum_{i=1}^{k} \|y_i - x_i WW'\|^2$$

对于每一个 one-hot 编码进行反向梯度传播，我们最终可以得到一个损失最小的函数。从逻辑上看即使用这个映射保证了相似 path 词语的词向量具备更近的余弦相似度。

3. 基于余弦相似度的 K-means

与普通的 K-means 相同，仅仅是聚类的时候使用了余弦相似度，以及外部添加词语的时同样使用余弦相似度进行簇类的估计。

余弦相似度的聚类方法如下（对于一个簇 n 个点向量 x，我们默认得到的向量范数为一）：

$$argmin_t \sum_{i=1}^{n} \frac{x_i^T \cdot y_i}{\|x_i\|}$$
$$s.t. \|y\| = 1$$
$$y = \frac{\sum_{i=1}^{n} \frac{x_i}{\|x_i\|}}{n}$$

最终可以根据设定的 K 值进行聚类。

四、算法及创新：

1.代码：代码由于形式并且比较复杂且较多，所以给出 github 链接：

https://github.com/Randle-Github/Machine-Learning-Experiment/tree/main/experiment3

其中 K-means 部分在 kmeans.py 为本次实验 K-means 核心。

2.个人创新：

1.创作形式：使用了标准的深度学习框架 pytorch，以及建立了两个作者合作的 repo 结构关系。

2.任务方法：引入了 NLP 中的 Word embedding 技术，将 K-means 这种简单的机器学习方法引入了复杂的自然语言处理任务。

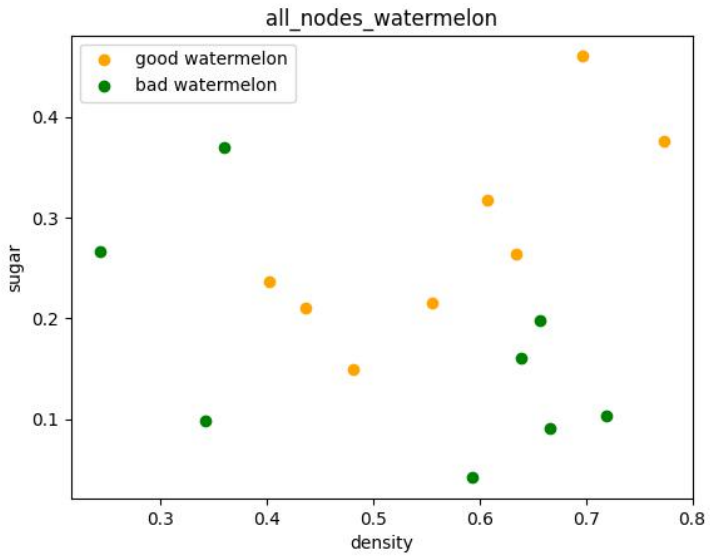
五、实验数据及结果分析：

由于代码量较大，所有代码存在 github 仓库中，链接如下：

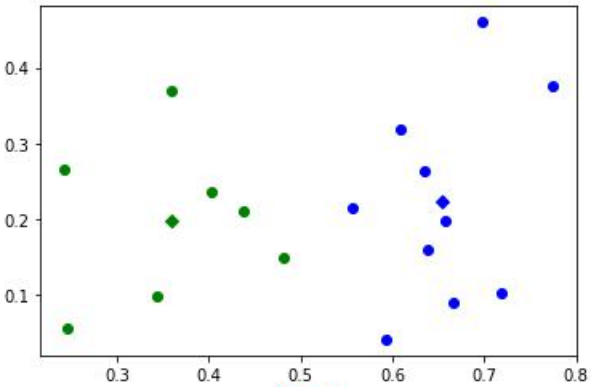
https://github.com/Randle-Github/Machine-Learning-Experiment/tree/main/experiment3

1. 西瓜数据集

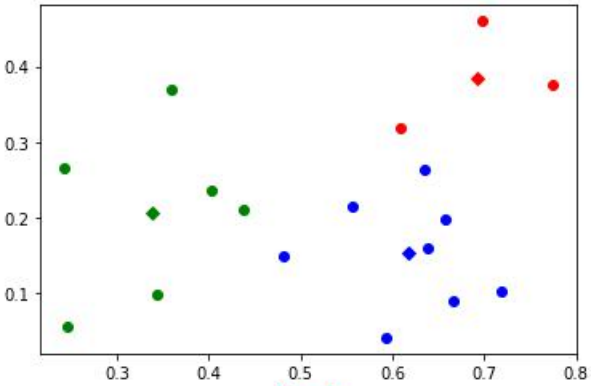
使用了简单的 KNN 算法对于西瓜数据集进行分类。由于数据量太少，并且只有二维，首先进行可视化：



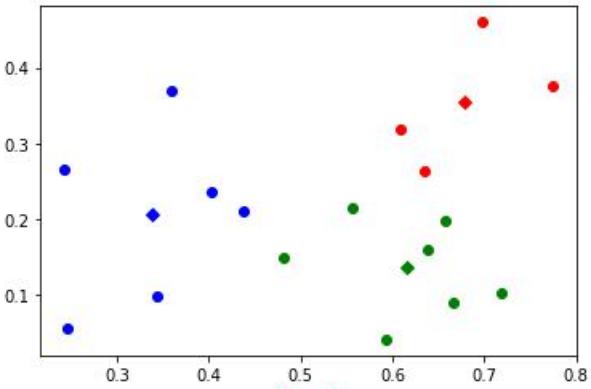
发现数据分布也很简单，对于不同的 K 进行测试，结果如下：


0.4
0.3
0.2
0.1
0.3
0.4
0.5
0.6
0.7
0.8

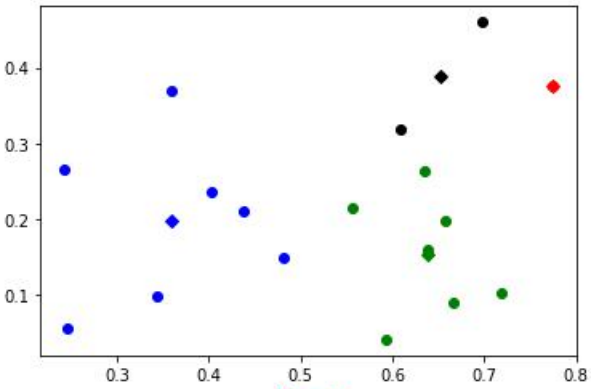
K=2


0.4
0.3
0.2
0.1
0.3
0.4
0.5
0.6
0.7
0.8

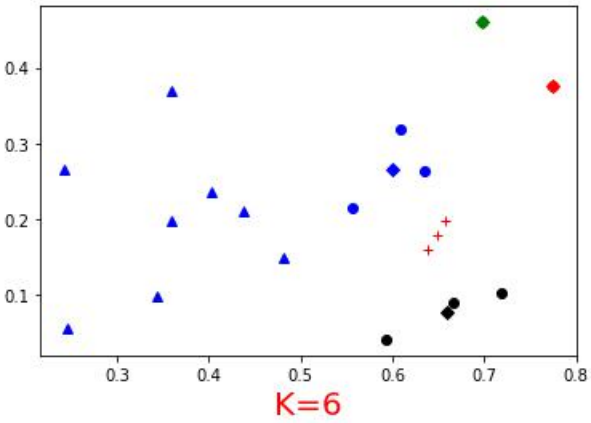
K=3

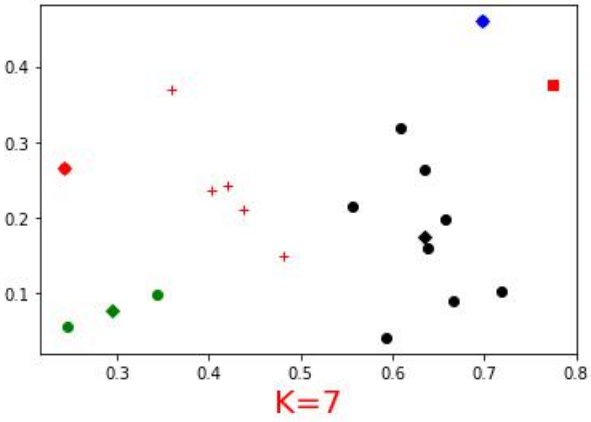

0.4
0.3
0.2
0.1
0.3
0.4
0.5
0.6
0.7
0.8

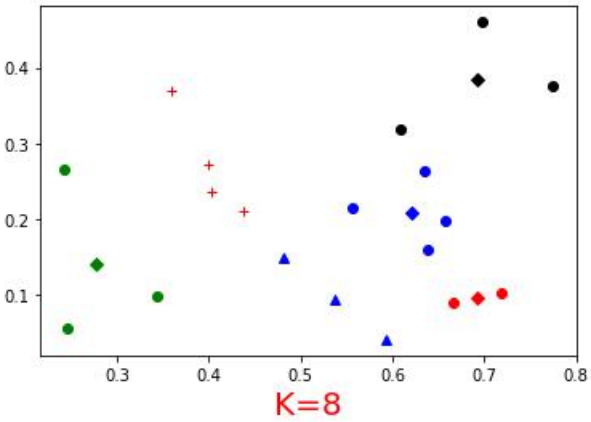
K=4


0.4
0.3
0.2
0.1
0.3
0.4
0.5
0.6
0.7
0.8

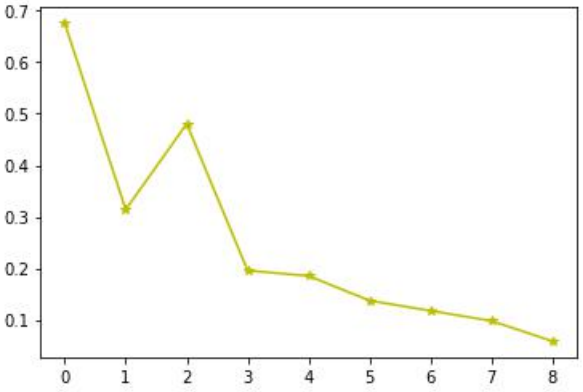
K=5







为了判断效果，我们引入 WSS 方法：



在 K=3 的时候聚类效果最好。

2. Word Clustering

我们的 repo 包含了：

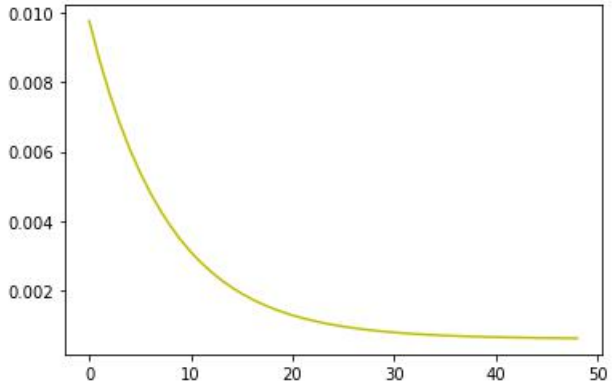
```
DeepWalk.py 深度游走部分
kmeans.py 最后一层的kmeans部分
node embedding.py node embedding深度学习部分
out.npy 词袋
main.py 数据预处理部分
```

由于我们做的只是一个简单的 demo 实验，所用的数据集使用的 500 字的小型文本，最终的结果噪声比较严重。

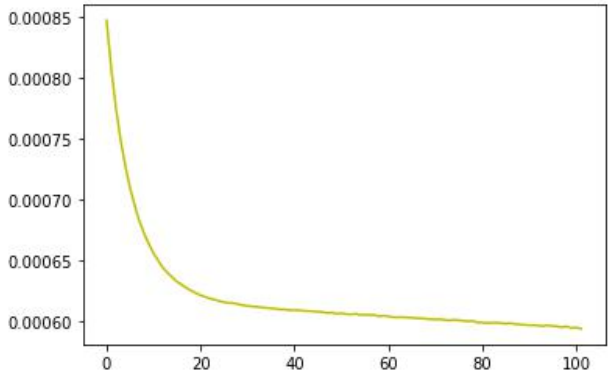
首先我们将文本在 main.py 中进行分词、词语与 one-hot 编码映射以及建图，将图输入 DeepWalk.py，我们得到了 path.npy，即词语路径。后续将词语路径输入 node_embedding.py 对于词语 path 进行了词袋的构建，结果存在 out.csv 中。

我们的深度学习过程 loss 变化如下：

这个是 embedded 词向量长度为 2 时的 loss 变化情况：



这个是 embedded 词向量长度为 100 时的 loss 变化情况：



可以看到映射到 2 维空间也能有较小的信息损失，但是在保留更高维度的向量时信息损

失更低。

最终得到的词向量部分展示：

-0.004361235070973635 -0.01387453731149435

-0.08038820326328278 -0.02150185965001583

-0.00888780690729618 0.004422661382704973

0.006702574901282787 0.004406054504215717

0.014521915465593338 0.04179384931921959

0.018933510407805443 0.022150203585624695

0.025583667680621147 -0.016417337581515312

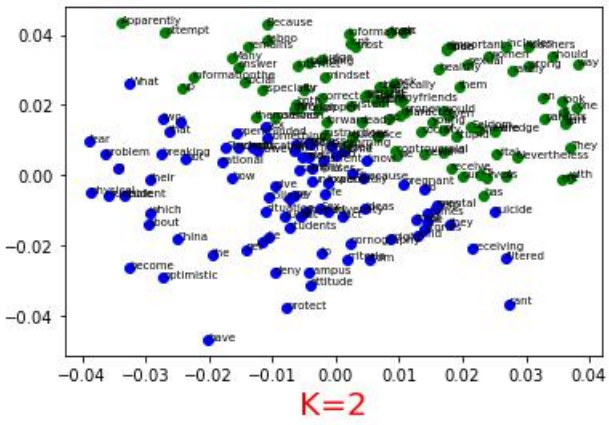
-0.030187079682946205 -0.058424100279808044

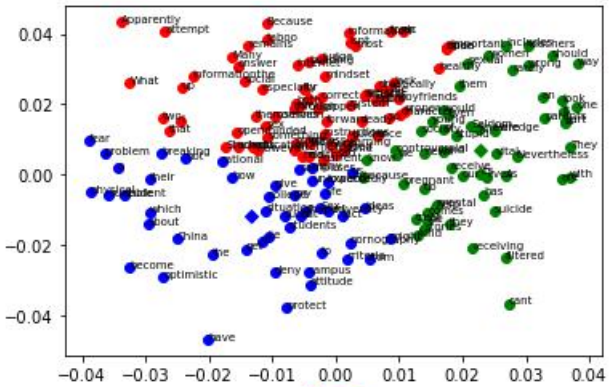
0.034372519701719284 0.0720856785774231

-0.14515739679336548 -0.08153604716062546

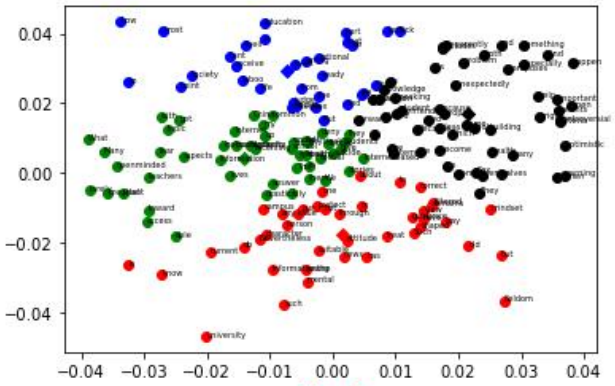
0.0075412532314658165 -0.05542005971074104

我们的结果可视化如下(对于 K-means 中 K 取值不同对词分类的影响,如从 K 从 2 到 8)：

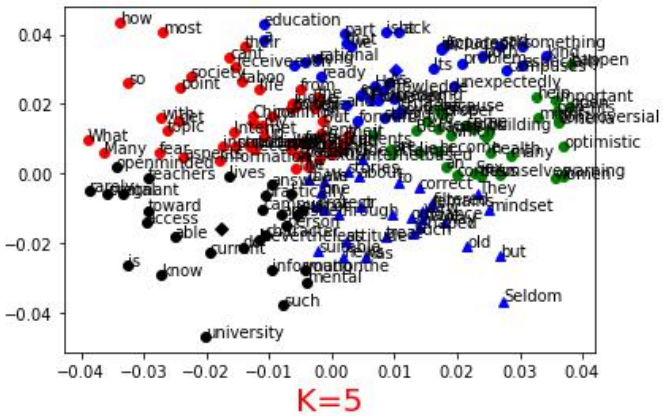



0.04
0.02
0.00
−0.02
−0.04
−0.04 −0.03 −0.02 −0.01 0.00 0.01 0.02 0.03 0.04

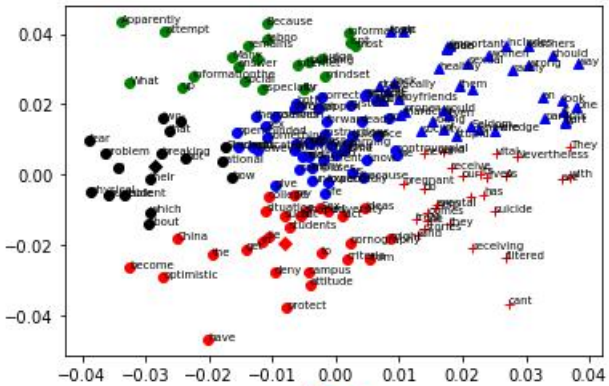
K=3


0.04
0.02
0.00
−0.02
−0.04
−0.04 −0.03 −0.02 −0.01 0.00 0.01 0.02 0.03 0.04

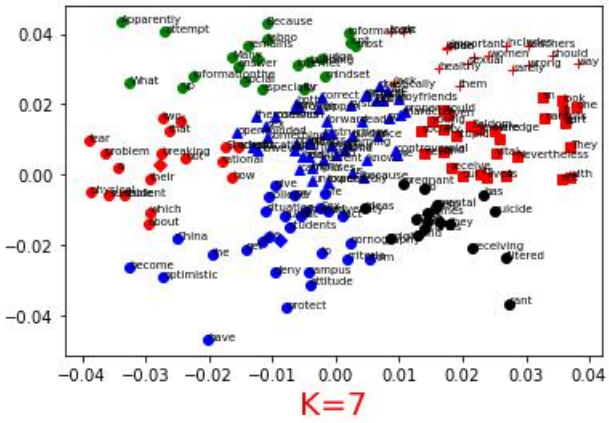
K=4

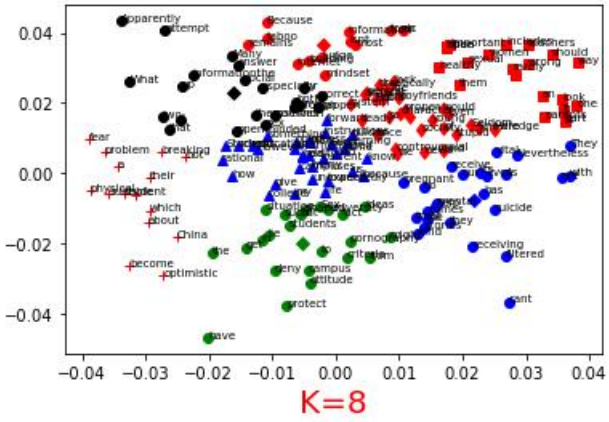

how
most
education
so
point
What
Many
fear
lives
toward
access
able
current
is
know
such
university
Seldom
but
old
optimistic
correct
they
mindset
0.04
0.02
0.00
−0.02
−0.04
−0.04 −0.03 −0.02 −0.01 0.00 0.01 0.02 0.03 0.04

K=5


0.04
0.02
0.00
−0.02
−0.04
−0.04 −0.03 −0.02 −0.01 0.00 0.01 0.02 0.03 0.04

K=6





可以看出，整个聚类结果是可行的。如果在更大的文本数据集可以得到很好的带有 semantic information 的 embedding 映射。比如在整本《哈姆雷特》进行训练，最终对人物词汇进行可视化的结果各个人物之间的关系可以清楚地表明。

六、总结及心得体会：

体会最深刻的一点是：对于 K-means 这种无监督学习方法，最重要的部分其实是 K 的选择以及 metric 方式。对于简单的数据集，仅仅使用他们的数据集本身的性质使用某种范数就可以达到很好的效果，但是对于文本数据集这种复杂空间，需要将数据分布映射到另外的一个低维度、信息密度更大的空间进行操作。

K-means 虽然是一种较为早期的机器学习方法，并且效果很大的受到了噪声、K 的选取以及 metric 方法的影响，但是与深度学习结合，仍然具有自己的用武之地。其在可拓展性部分具有极大的优势，在后续研究的阶段，K-means 的思想十分简单但是不可忽视。

七、对本实验过程及方法、手段的改进建议：

改进建议暂无。

报告评分：

指导教师签字：