【NLP作业-04】词向量模型及聚类分析

杨思捷　ZY2103533

1. 引言

本次作业使用Word2Vec生成词向量并对词向量进行分析，首先尝试使用Pytorch搭建CBOW模型，发现效果不好后使用`gensim`的词向量模型。

2. 文本的表示

2.1. One-hot

可以使用一个词表大小的向量来表示一个词，例如有一个词表$V = \{今天，天气，真好\}$，则可以用一个大小为3的向量表示今天这个词，$e_i = [1, 0, 0]$。

通过One-hot表示词时，向量的大小由词库决定，对于庞大的语料库而言会引发维度灾难；并且它无法度量词语之间的相似性。因而引出了词向量的方式表示。

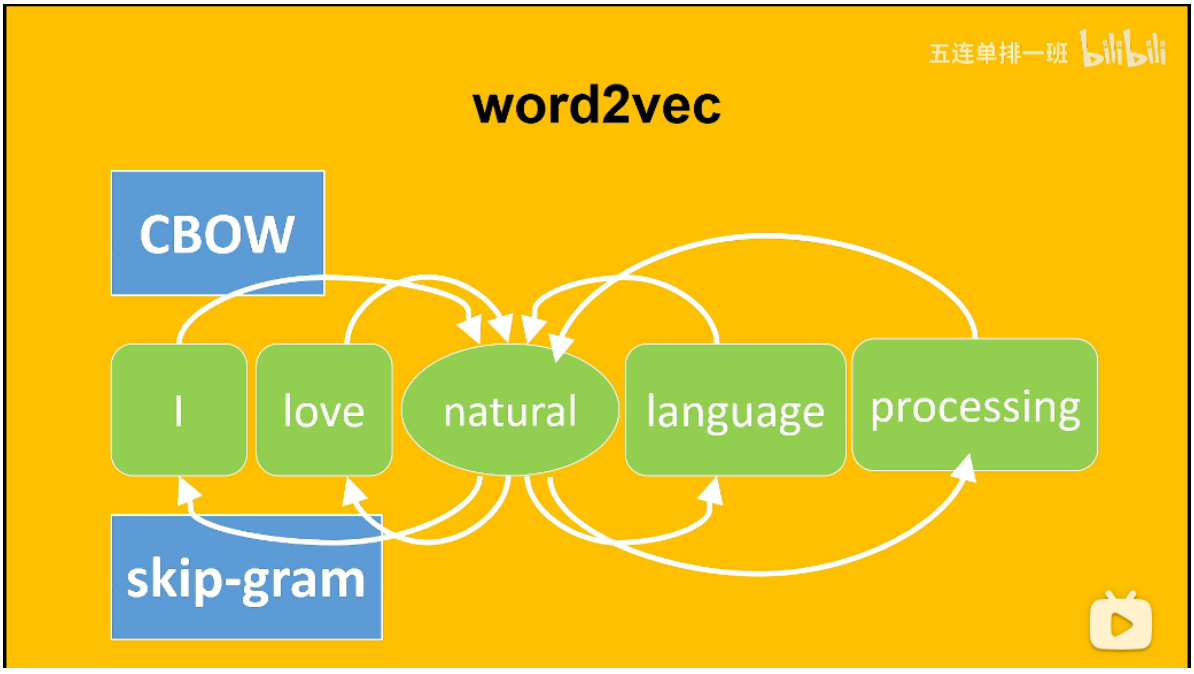
作业1中计算中文的信息熵的时候使用了一元模型、二元模型，这些是词的**分布式模型**。词的分布式表示中，可以使用一个连续、稠密、低维的向量表示词，但向量值通过对语料库进行统计后，经过点互信息、奇异值分解等变换后就无法更改了。而**词向量**中的向量值会随着目标任务的优化自动调整，但是目标任务训练数据较少时，学习合适的词向量难度会比较大。

2.2. Word2Vec

词向量的训练方法有许多种，本作业中使用Word2Vec。

Word2Vec中有两种模型，CBOW(Continuous Bag-of-Words)和Skip-gram模型。CBOW是用上下文的词语预测中心词，而Skip-gram则反过来用中心词预测上下文词语。

给定一个窗口大小然后指定上下文词语的范围，最大化中心词和上下文词语同时出现的概率即可得到词向量。



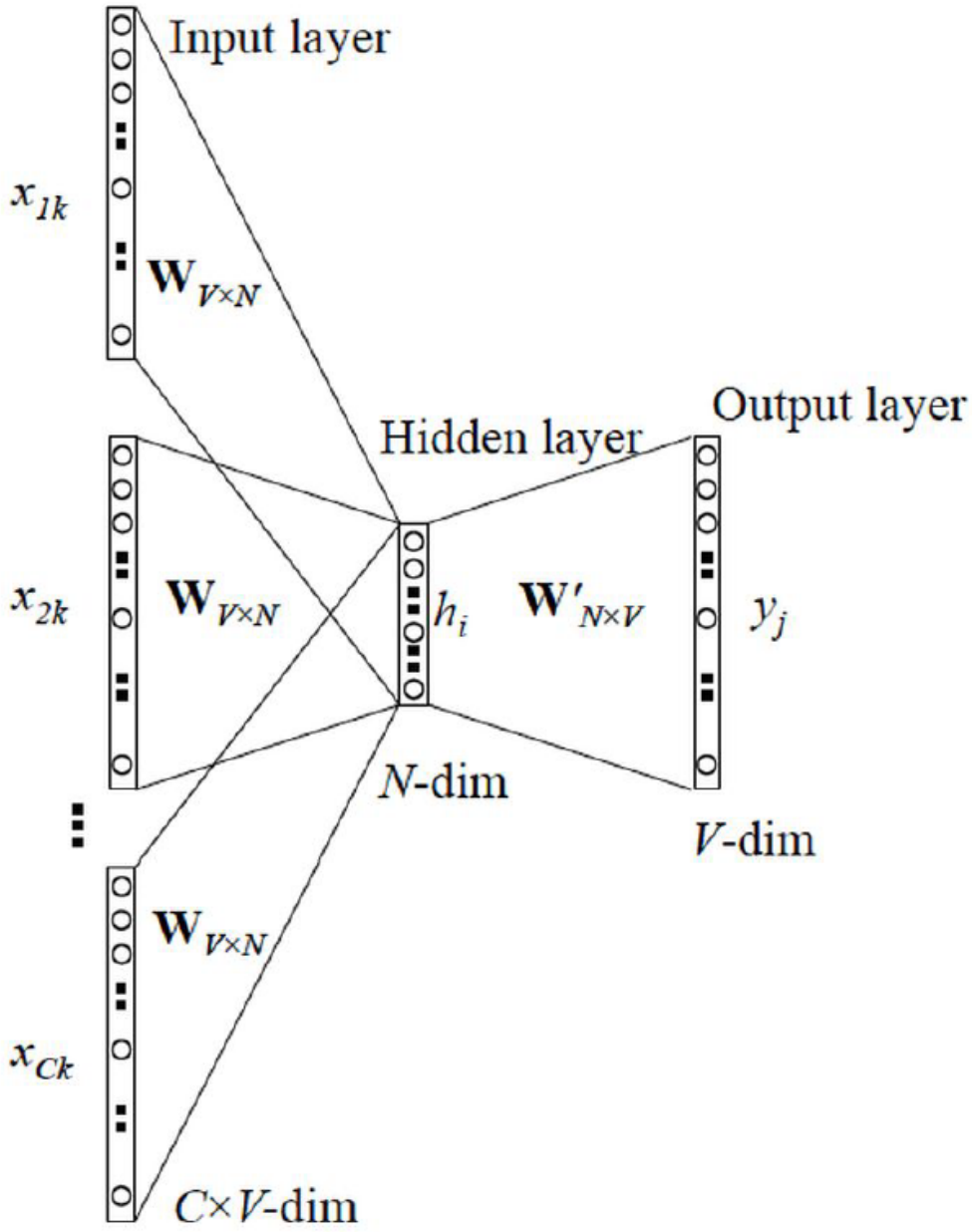
如何评估词向量提取特征的效果无法直接从词向量的数值表达中体现，因而可以通过给定一个词语输出相似词、或者给定两个相似词计算其相似度、以及将数据可视化进行评估。

Word2Vec也有一些**局限性**，无法处理同义词，Word2Vec模型的文本和词向量是一一对应的，无法根据上下文分析其具体的语义；窗口长度有限；由于使用了词袋模型，语序不是特别严格。

3. 词向量模型

3.1. CBOW模型

CBOW模型如下图所示：



CBOW的输入是一个词语的上下文，输出是这个词语。数学语言描述为，对文本 $\mathbf{w}=\cdots w_{t-2}w_{t-1}w_tw_{t+1}w_{t+2}\cdots$，当上下文窗口大小为5时，上下文为 $C_t=\{w_{t-2},w_{t-1},w_{t+1},w_{t+2}\}$，需要通过上下文$C_t$预测单词$w_t$。

输入层：当上下文窗口大小为5时，在目标词周围取2个词作为模型的输入，通过4个维度为词表长度的One-hot向量表示，记为e_{w_i}

词向量层：词向量可以从由One-hot向量经过权重矩阵$E\in\mathbb{R}^{|V|\times d}$映射而来，$v_{w_i}=Ee_{w_i}$。其中$V$是语料库的词表，d是隐含层的层数

输出层：需要输出的结果为词w_t，需要将C_t中的词向量取平均$v_{C_t}=\frac{1}{|C_t|}\sum_{w\in C_t}v_w$，然后通过矩阵$F\in\mathbb{R}^{d\times|\mathbb{V}|}$映射而来，$w_t=Fv_{C_t}$。

其中，矩阵E和矩阵F是需要训练得到的权重矩阵。

训练：通过迭代，需要获得使得CBOW模型预测w_t概率最大的权重E。已知上下文C_t，输出单词w_t的概率可以通过下面的公式计算，其中$v^{'}_{w_t}$是矩阵F中，单词w_i对应的列向量。

$$P(w_t|C_t)=\frac{\exp\left(v_{C_t}\cdot v^{'}_{w_t}\right)}{\sum\limits_{w_i\in V}\exp\left(v_{C_t}\cdot v^{'}_{w_t}\right)}$$

当给定一段长为T的文本时，损失函数有下面的形式

$$Loss=-\sum_{i=1}^{T}\log\left(P(w_t|C_t)\right)$$

3.2. 词向量的相关性

通过上面的CBOW模型计算得到词向量之后，可以计算空间内的余弦相似度来度量两个词的相关性

$$sim(w_a,w_b)=cos(v_{w_a},v_{w_b})=\frac{v_{w_a}\cdot v_{w_b}}{|v_{w_a}||v_{w_b}|}$$

4. 代码实现

对金庸的小说不是太熟，选用了相对了解多一点的倚天屠龙记作输入进行分析。

4.1. 预处理

根据上面的分析，需要先将语料库断句分词，并使用One-hot法生成词表。需要注意的是，由于jieba分词对专有名词的支持不是太好，需要补充专有名词字典。

这部分的核心代码如下

```
def build(self,corpus):
    '''
        从语料库中建立词表
        保存词表并保存用One-hot表示的语料库
    '''
    word2id_dict = {}
    id2word_dict = {}

    corpus_onehot = []


    word_id = 0
    jieba.load_userdict("user_dict.txt")
    for sentence in corpus:
        sentence_segL = jieba.cut(sentence)
        sentence_onehot = []
        for word in sentence_segL:
            word = word.strip()
            if word not in word2id_dict:
                word2id_dict[word] = word_id
                id2word_dict[word_id] = word
                word_id += 1
            sentence_onehot.append(str(word2id_dict[word]))
```

```
        corpus_onehot.append(",".join(sentence_onehot)+"\n")
    self.word2id_dict, self.id2word_dict = word2id_dict, id2word_dict

    with open("out/corpus_onehot.txt","w",encoding="utf-8") as fp:
        fp.writelines(corpus_onehot)

    with open('out/word2id_dict.json','w',encoding="utf-8") as fp:
        json.dump(self.word2id_dict, fp,indent=4,ensure_ascii=False)
    with open('out/id2word_dict.json','w',encoding="utf-8") as fp:
        json.dump(self.id2word_dict, fp,indent=4,ensure_ascii=False)
```

完整代码参见`utils.py`中的WordTable类。

4.2. 模型实现

需要构建CBOW的数据集和模型。

首先是CBOW数据集的构建，下面的代码是取了上下文窗口为5的例子

```
class CbowDataset(Dataset):
    def __init__(self, corpus, context_size=5):
        '''
            传入one-hot形式的语料库
        '''
        self.data = []
        context_size_half = int((context_size-1)/2)
        for sentence in tqdm(corpus, desc="Dataset Construction"):
            if len(sentence) < context_size:
                continue
            for i in range(context_size_half, len(sentence) -
context_size_half):
                # 模型输入：左右分别取context_size长度的上下文
                context = sentence[i-context_size_half:i] +
sentence[i+1:i+1+context_size_half]
                # 模型输出：当前词
                target = sentence[i]
                self.data.append((context, target))
```

CBOW网络模型的构建核心代码如下，主要是网络层的定义和前向传播

```
class CbowModel(nn.Module):
    def __init__(self, vocab_size, embedding_dim):
        super(CbowModel, self).__init__()
        # 词嵌入层
        self.embedding_layer = nn.Embedding(vocab_size, embedding_dim)
        # 隐含层->输出层
        self.output_layer = nn.Linear(embedding_dim, vocab_size)

    def forward(self, inputs):
        embeds = self.embedding_layer(inputs)
        # 对上下文词向量求平均
        hidden = embeds.mean(dim=1)
        output = self.output_layer(hidden)
        log_probs = F.log_softmax(output, dim=1)
        return log_probs
```

这部分的代码参见`CBOW.py`。

然后就可以通过训练集和和模型进行训练训练了，核心代码如下：

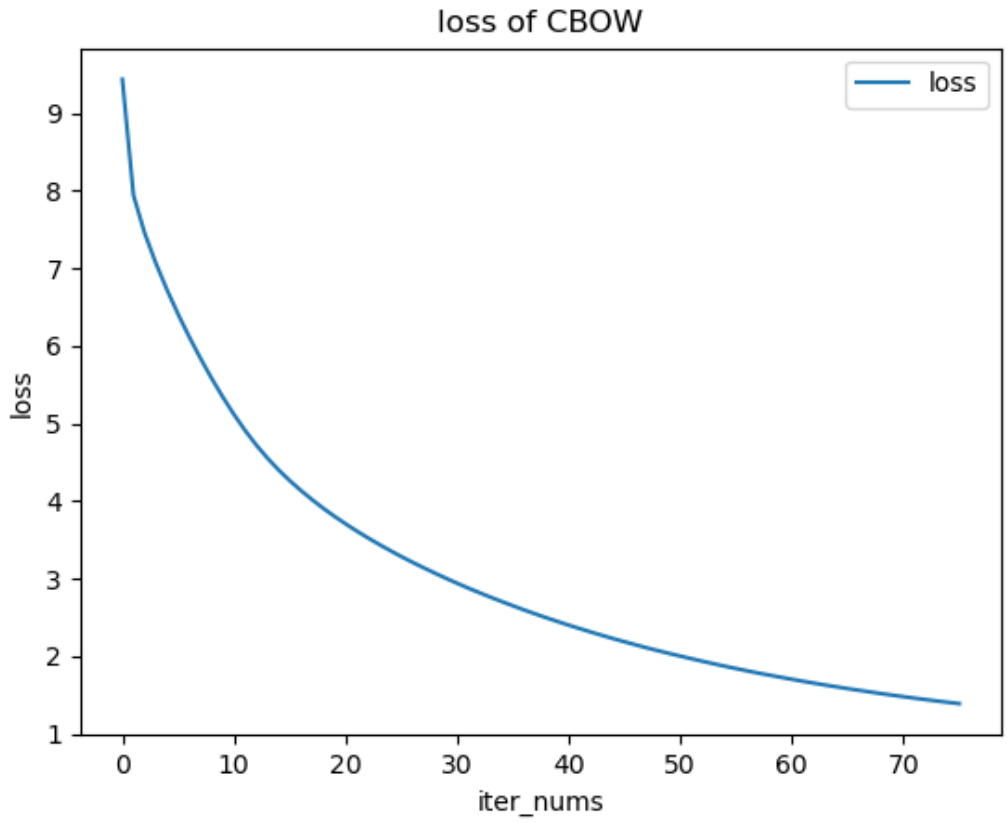
```
for iteration,[inputs, targets] in enumerate(pbar):
    inputs, targets = inputs.to(device), targets.to(device)

    log_probs = model(inputs)
    loss = criterion(log_probs, targets)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    total_loss += loss.item()
    val_loss = total_loss / (iteration + 1)
```

训练的完整代码参见`CBOW_train.py`，训练的损失函数图如下：



70次迭代后损失值较小。

5. 结果分析

尝试使用pytorch自己实现一个CBOW，但是效果很差，最后还是使用`gensim`的model了。

5.1. Pytorch搭建的CBOW

取两个人的不同称号计算相似度，比如金毛狮王谢逊、蛛儿殷离、白眉鹰王殷天正、青翼蝠王韦一笑

下面是取了迭代40次的结果计算出来的相似度，可以看出相似度不是太高

```
金毛狮王&谢逊                    similarity=0.066090
蛛儿&殷离                        similarity=0.080334
白眉鹰王&殷天正                  similarity=0.008188
青翼蝠王&韦一笑                  similarity=0.119304
```

取迭代70次的结果后相似度有所上升，

```
金毛狮王&谢逊                    similarity=0.102201
蛛儿&殷离                        similarity=0.082431
白眉鹰王&殷天正                  similarity=0.035346
青翼蝠王&韦一笑                  similarity=0.111895
```

但是效果也不是太好，此时回过头分析词向量生成的效果可能不是太好，使用`gensim`的模型。

5.2. 相似度分析

使用`gensim`模型重新分析两个人物的称号。

```
金毛狮王&谢逊                    similarity=0.647641
蛛儿&殷离                        similarity=0.892283
白眉鹰王&殷天正                  similarity=0.559445
青翼蝠王&韦一笑                  similarity=0.570128
```

5.3. 输出相近词

取张无忌、周芷若、屠龙刀、倚天剑输出最相近的10个词，可以看到结果如下：

	张无忌		周芷若		屠龙刀		倚天剑	
1	赵敏	0.927832	赵敏	0.889759	刀	0.784966	刀	0.798483
2	张翠山	0.907328	张翠山	0.868075	一起	0.772851	掌力	0.759898
3	俞岱岩	0.871139	张无忌	0.866964	成昆	0.732689	木剑	0.753553
4	宋青书	0.868416	纪晓芙	0.862152	倚天剑	0.72743	挥舞	0.744254
5	周芷若	0.866964	殷素素	0.850056	他们	0.722515	短剑	0.744012
6	蛛儿	0.86589	小昭	0.847232	取	0.715734	狼牙棒	0.739048
7	殷素素	0.864154	宋青书	0.843286	谢逊	0.713947	妻子	0.738043
8	金花婆婆	0.850569	蛛儿	0.832284	公子	0.711392	金元宝	0.732128
9	朱九真	0.845222	都大锦	0.82892	你们	0.706641	两根	0.729098
10	谢逊	0.836777	朱九真	0.827019	七伤拳	0.70452	针刺	0.729023

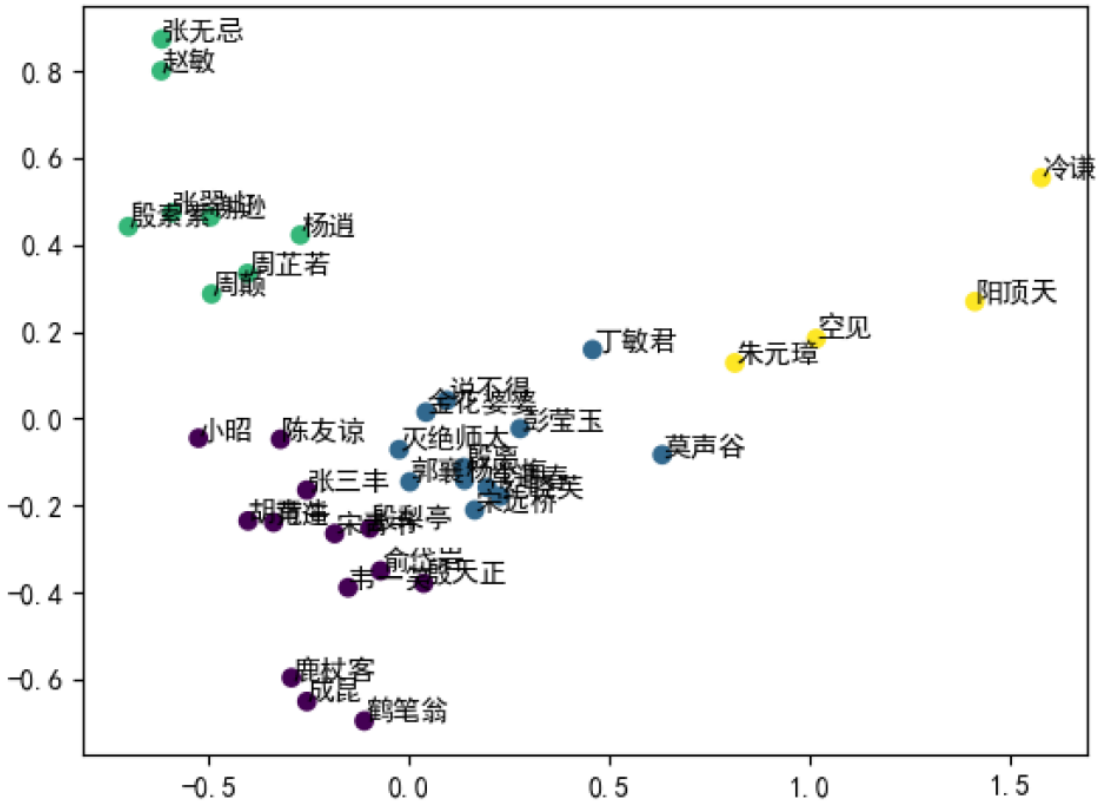
5.4. 聚类分析

取一系列词语进行聚类分析：

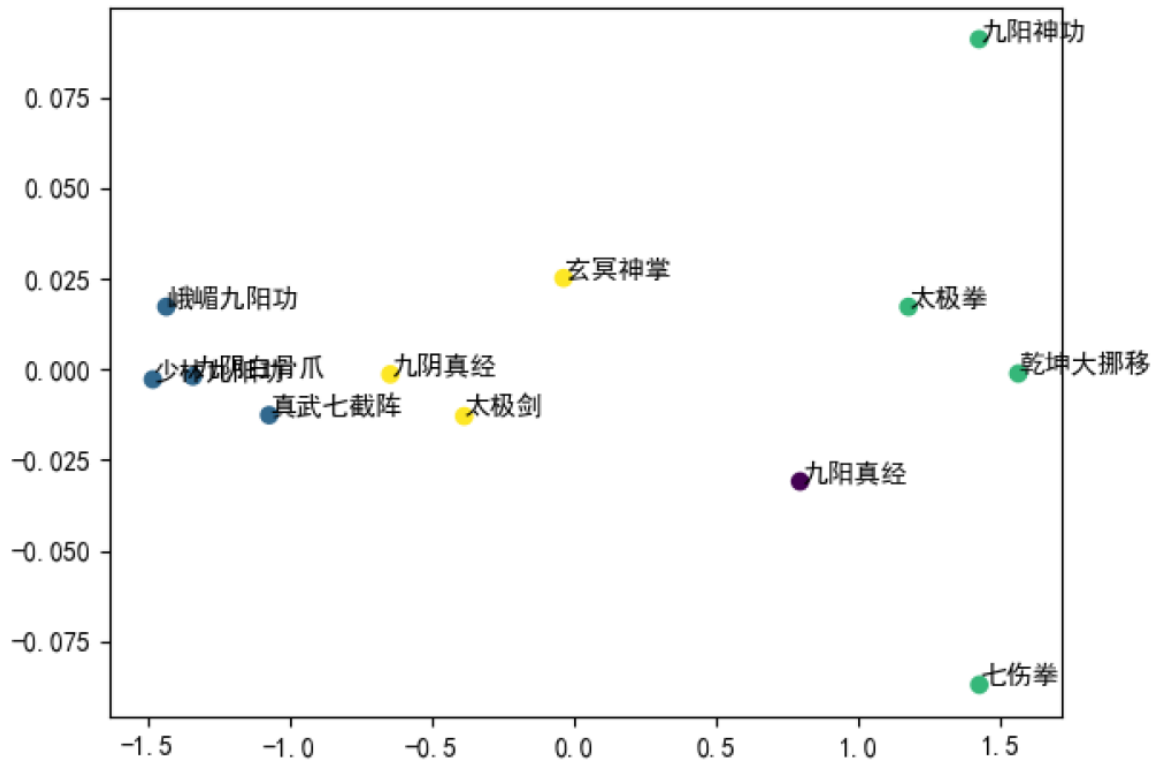
```
Character = ['张无忌', '赵敏', '谢逊', '周芷若', '张翠山', '杨逍', '灭绝师太', '张三丰', '殷离', '殷素素', '小昭', '殷梨亭', '韦一笑', '俞岱岩', '胡青牛', '宋远桥', '殷天正', '周颠', '纪晓芙', '宋青书', '杨不悔', '金花婆婆', '说不得', '范遥', '丁敏君', '陈友谅', '常遇春', '鹿杖客', '莫声谷', '鹤笔翁', '朱元璋', '阳顶天', '郭襄', '成昆', '空见', '冷谦','彭莹玉']
Kungfu = ['太极拳', '太极剑', '真武七截阵', '九阳神功', '峨嵋九阳功', '少林九阳功', '乾坤大挪移', '九阴真经', '九阴白骨爪','七伤拳', '九阳真经', '玄冥神掌']
sect = ['明教', '峨嵋派', '天鹰教', '昆仑派', '崆峒派', '武当派', '少林派','神拳门', '华山派']
```

取人物，功法，门派三类进行聚类分析

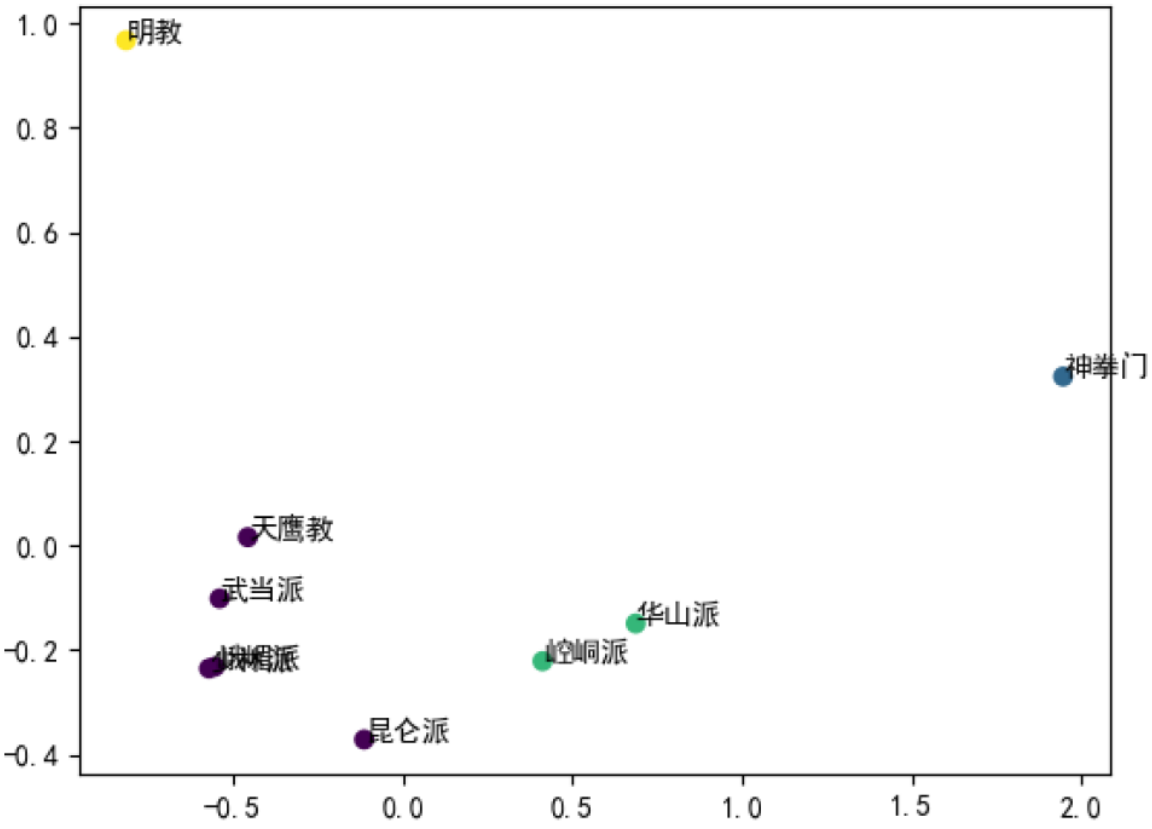
人物关系如图，可以看出张无忌和赵敏的纠葛和爱恨情仇比较多，关系就离的比较近。



武功关系如下，因为没怎么看过小说，感觉分类还算恰当？



门派关系如下图：



6. 总结

由于jieba分词对一些专有名词处理的的不够细致，所以最后的结果会不太好。比如七伤拳就被分割成了七伤和拳两个单词，玄冥神掌被分割为了玄冥和神掌两个单词。

通过CBOW的方法生成词向量的方法可以获得词与词之间的相似度，通过词向量提取文本特征，为后续的一些NLP任务提供了基础。但是在本次作业中，我搭建的CBOW模型可能存在一些问题，提取

7. 参考资料

https://www.bilibili.com/video/BV1vS4y1N7mo

https://github.com/bamtercelboo/pytorch_word2vec

https://www.cnblogs.com/peghoty/p/3857839.html

https://github.com/HIT-SCIR/plm-nlp-code