# 中国科学院大学
University of Chinese Academy of Sciences

# 自 然 语 言 处 理

## 技术报告

**班级：**2023 春季自然语言处理班

**组号：**--

**姓名：**唐嘉良

**学号：**2020K8009907032

**报告主题：**基于 NNLM 和 word2vec 的中文词向量计算

**2023 年 5 月 5 日**

# 一．报告摘要

本次实验中，我基于 n-grams 语言模型理论，利用 FNNLM、word2vec 开源模型和 tensorflow 深度学习框架等技术，对北京大学 1998 年人民日报分词语料库进行中文词向量计算，并对计算结果进行分析，给出结论与思考。

一方面，我们分别采用 C++和 Python 语言实现的 FNNLM（前馈神经网络语言模型）工具进行中文词向量计算，它们分别是 http://nlg.isi.edu/software/nplm/和 https://github.com/FuYanzhe2/NNLM。选取合适的参数以达到更好的效果，最终发现前者在核心参数 embedding dimension = 100 以及 vocabulary size = 6000 时达到较好效果，但训练代价较高；而后者训练时间短，但由于自身网络复杂程度限制而在词向量相似度检查中表现欠佳。

另一方面，我们采用开源 word2vec 工具 https://github.com/svn2github/word2vec。选取合适的参数，在同一语料库下训练，将其与 FNN 语言模型进行全方位的对比，最终发现该 word2vec 在训练代价上远低于 FNNLM。在训练效果上，word2vec 由于作了数据平滑化处理，在低频词的近义词匹配上效果远好于 FNNLM，代价是在高频词的近义词匹配上丢失了部分准确性。

# 二．FNNLM

1.C++实现（开源模型）

采用开源模型，该模型具有两层 FNN 网络，接受输入为已分词的文本文件，输出前馈神经网络训练结果（包含词向量结果与网络参数）。

我们首先按照作业要求，选定 3-grams 语言模型，词向量维度 10，词汇表大小 1000，训练 epoch=10，采用均匀分布初始化，并将其余词汇全部用<unk>代替，训练得到模型。随机选取词汇计算与其余弦相似度最高的词，部分结果如下（第一行为选定词，随后是与之最相似的词汇及其相似度）：

| 公报 | 工厂/n |
| --- | --- |
| ('释放', 0.44396305967286204) | ('正在/d', 0.43871050792997535) |
| ('结合', 0.4303669442497134) | ('公布/v', 0.43348676969449834) |
| ('工业', 0.42874544834252304) | ('上百/m', 0.4136978817568357) |
| ('寄予', 0.4125484729024546) | ('电视剧/n', 0.3695310965806779) |
| ('一九九一年', 0.39168772402986235) | ('官员/n', 0.34788182206552626) |
| ('行动', 0.3630249513607826) | ('解困/v', 0.34617478068740604) |
| ('背后', 0.35373269160320187) | ('最佳/z', 0.34430927415405543) |
| ('中共', 0.3478142831377674) | ('５０００/m', 0.3431802268289895) |
| ('职位', 0.34451555702862025) | ('贫困/a', 0.34183379141858755) |

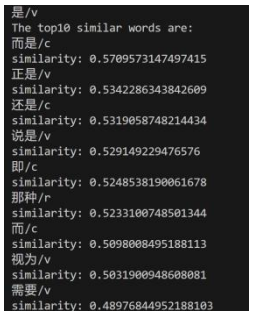
“公报”与“释放”“结合”、“工厂”与语义上相距甚远。可见模型训练效果很差。猜测是因为词向量维度过低，所能表示的差异化信息不足。另外，还有可能是因为词汇表过小，有一些词义相近但是出现频率不高的词汇没有被列入，导致匹配不到该词汇。

将词向量维度升级为 100，词汇表大小扩张成 6000，其余参数维持不变，再次训练模型。并随机选取词汇计算与其余弦相似度最高的词，部分结果如下：

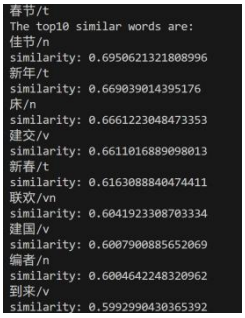


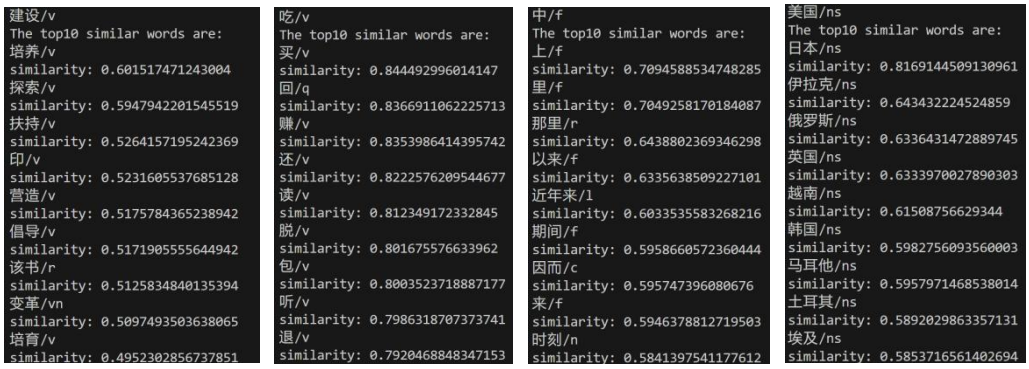
其中“香港”作为地区词，与“成员国”“苏南”等地区词语义接近，可见参数调整初见成效，验证了我们之前的想法。但是模型发挥不稳定，例如“中国”一词的匹配结果不如人意，甚至匹配到的同一词性的词语都很少，但观察输出我们可以发现一个有意思的事实：词语“中国”的匹配结果似乎都是经常与其一起出现的词，例如“廉洁自律”“禁毒”“攻坚”，但本身语义却不一致。

考虑到这一点，结合模型的不稳定性，猜测是由于 3-grams 语言模型窗口大小太小，训练信息比较局部与片面，导致训练出来的词向量中包含的词语关联信息有偏差，为了获取更长的词汇出现历史，我们将模型调整为 5-grams，其余参数不变。再次训练，随机选取 20 个词汇，计算匹配与其最接近的 10 个词汇，结果如下：








从上面的结果中可以看出，该模型训练效果很好，在大多数随机选取的测试用例上表现良好，例如“提高”匹配“增强”“扩大”“改善”等。这说明此前性能瓶颈确是因为3-grams语言模型的缺陷。另外，还有个别测试用例表现不佳，例如“天空”和“年轻人”，匹配到的词汇要么毫不相干，要么甚至语义相反（例如“年轻人”与“老伴”“奶奶”），在输出文件中查找得知，这一类表现不佳的词汇大部分来自词汇表尾端词汇，在语料库中出现频率太低，因此训练数据缺乏可靠性；另一方面，有些词诸如“执法”没有太多常见的近义词（即词表中该词的近义词收录较少，且出现频率均较低），所以关于其近义词词

向量的训练样本也不足，所以即便有较高的相似度，也依然存在结果的偏差。

2.Python 实现（开源模型+Tensorflow 框架）

采用开源模型，该模型是对 2003 年 Yoshua Bengio 的 FNNLM 论文的简单复现，它接受输入为已分词的文本文件，输出前馈神经网络训练结果（包含词向量结果与词汇表）。

该 FNN 仅有一层网络层，受 C++版本模型的结果启发，我们放弃低维词向量和小词汇表，直接将词向量维度设置为 100，词表大小为 6000，采用 5-grams 模型，训练 epoch=10，采用均匀分布初始化和 Adam 优化器，并设置梯度阈值，隐藏层神经元数目为 128。

训练得到模型。随机选取词汇计算与其余弦相似度最高的词，部分结果如下：



可以发现模型训练效果不如人意，在正确性上与 C++模型天差地别。唯一的优势是该模型训练速度极快，对北大 1998 分词语料单线程训练 1 次仅需半分钟，但对于同样的文本、模型参数和训练次数，C++模型在开启 10 线程的情况下训练一次需要 8 分钟。说明该模型网络复杂程度太低，需要加深网络层数以寻求性能的突破。这也提示我们需要深层的网络和高维的向量才能有效且充分地挖掘文本信息，尽管这会带来较大的计算资源开销。

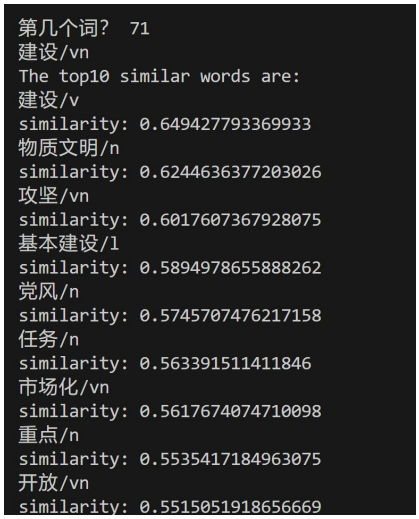
最后是关于词表大小的结论，如果词表设置过小，很可能丢失掉大量的词汇信息，以至于匹配效果差，然而词汇表过大会带来大量低频词汇的词向量训练，对计算资源的要求增加，而且低频词样本过少，会导致低频词词向量训练不充分，可靠性过低，从而在相似度匹配中造成错误的判断。

## 三．Word2vec

利用开源模型，该 word2vec 工具主要包含两个模型：跳字模型（skip-gram）和连续词袋模型（continuous bag of words），以及两种高效训练的方法：负采样（negative sampling）和层序 softmax（hierarchical softmax）。

将该模型的参数设置为 5-grams，词向量维度 100，词汇表约为 6000，采用 CBOW 模型

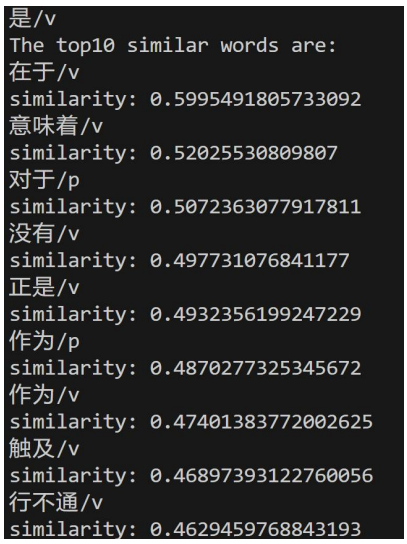
和负采样方法，训练得到模型。随机选取词汇，进行词向量相似度计算，选取一部分结果如下：

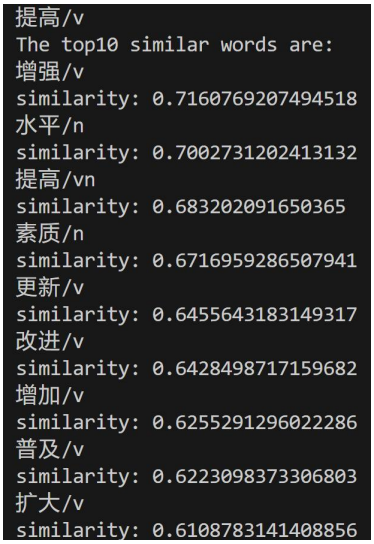


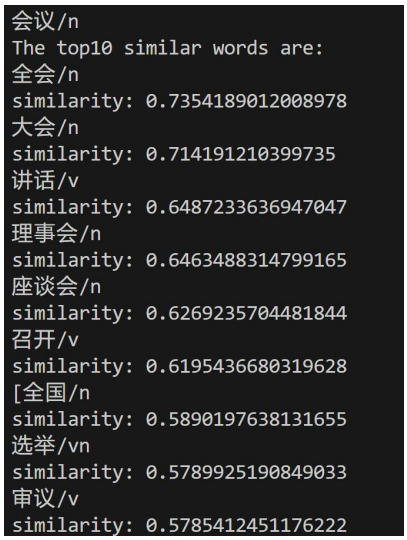


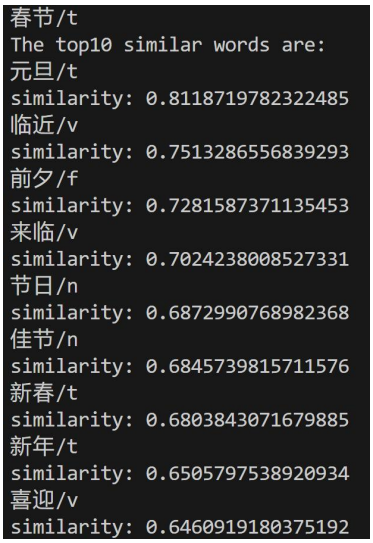
可以看到，该模型具备一定的效果，但是不及 FNNLM 的效果显著。针对该问题，我们尝试更大程度利用计算资源，调整词向量维度为 200，其余参数不变。总训练时长约为 20s。

为方便与 FNNLM 模型对比，我们选取了同一批词汇，挑选多个高频词与低频词代表，计算词向量。高频词结果如下：

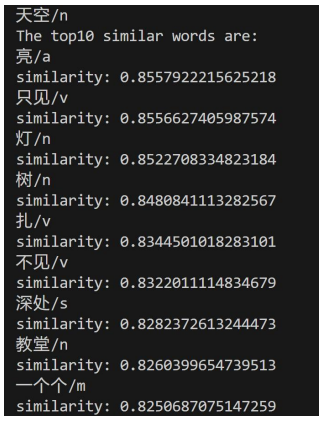


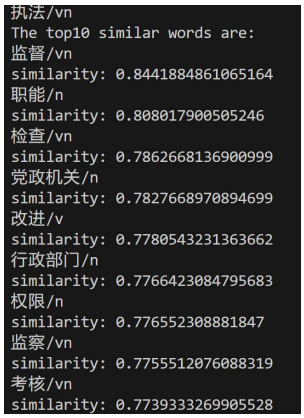



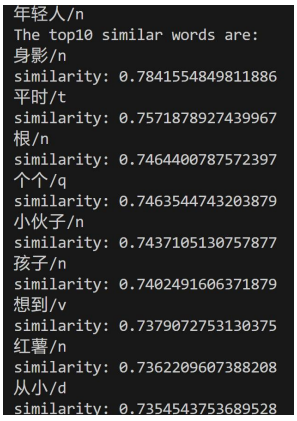



在高频词的匹配效果上面，word2vec 发挥稳定且可靠，这进一步证明高频词丰富的语料可以带来其词向量训练效果的显著。但是某些高频词如“是”，其匹配效果相比 FNNLM 来说并不更加卓越，前几位的匹配结果中掺杂的非近义词更多。为更为全面地比较两种模型，我们计算低频词结果如下：




从结果上可以看出，对于语料库中的超低频率词汇（如“天空”，位列词汇表 5000 位-6000 位，出现频率不足 30），训练效果同样不尽如人意。但是对于亚低频词汇如“执法”“年轻人”，其匹配效果远好于 FNNLM，匹配结果语义相关性更高。

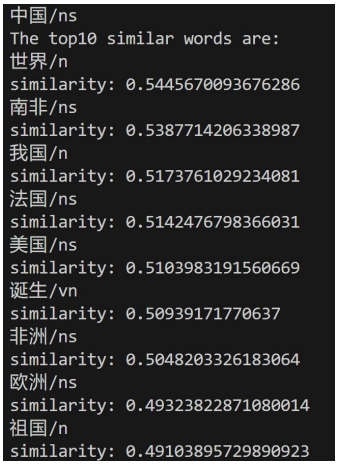
## 四. FNNLM v.s. Word2vec

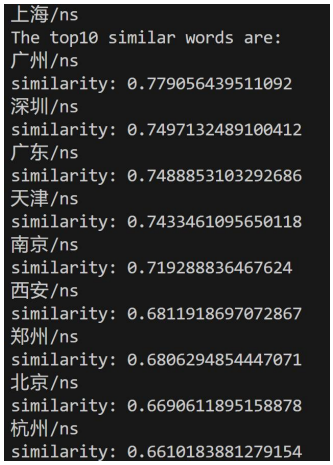
我们计算了不同频率词汇匹配结果，并对两种模型进行了对比。word2vec 在低频词的近义词匹配上效果远好于 FNNLM，代价是在高频词的近义词匹配上丢失了部分准确性。而对于超低频词汇，二者的训练效果均不佳。出现这一差异的的原因应当是由于该 word2vec 模型对高频词语料的出现频率设置了遏制机制，并对低频词语料进行频率强化，在保证高频词的训练效果下降低频词的训练精度损失，从而使得训练模型在不同频率词汇上更加平滑，属于一种“公平机制”。在该机制下，尽管我们会损失一些高频词匹配精度，但是低频词匹配精度会得到质的增强。

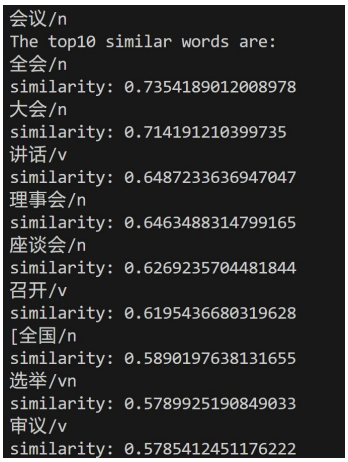
此外，在训练代价上，相似参数的两种模型训练代价天差地别。首先，二者的源码并不一致，FNNLM 采用 C++与 python(with pytorch)，而 Word2vec 采用的是 C 语言，语言自身的差异导致了训练代价的不同，Word2vec 的训练耗时远小于 FNNLM 的两个模型。

并且，在 FNNLM 的训练中，发现训练时间对词汇表大小比较敏感，这在一定程度上限制了词汇表的扩充。而且 FNNLM 在词汇表很大的时候性能急速下降，受低频词干扰严重。但是 Word2vec 的训练精度对词汇表大小并不敏感：在其他参数不变，调节词汇表大小从~6000 到~17000 的过程中，模型表现十分稳定，近义词匹配任务的准确性并未受到明显影响。

最后，我们发现模型训练结果与语料库自身有密切联系。由于我们选取的是北京大学 1998 年人民日报分词语料，其语言较为正式与官方，因此训练出来的模型在国家与地区名等书面词汇上表现极好，例如 FNNLM 模型（C++）对“中国”的匹配结果，和 Word2vec 对“上海”和“会议”的匹配结果：




这说明在训练模型时应当使用全面且充足的语料库，这是模型训练精度提升的基础。