基于深度学习的文章推荐实验报告

谭新宇 2016010649　　周泽龙 2016013231　　刘昀瑞 2016011134

实验大纲

摘要：本次试验中首先我们对所有文章进行预处理，然后再通过 TF-IDF 算法提取出每篇文章的 12 个特征词代表此文章，接着再通过 Word2Vec 模型进行基于 Skip-Gram 模型的三层神经网络的训练，然后再为每位用户构建特征，最后在测试文件中找到最有可能相关的 5 篇文章。

关键字：TF-IDF 特征词 Word2Vec 词向量 余弦距离

实验大致流程

预处理->深度学习计算词向量->推荐文章

实验具体分析

一、预处理

由于老师上课时的课件上曾经说过 TF-IDF 在实验中对于文本推荐”work well”,所以一开始我们就想直接利用 TF-IDF 频率向量作为文章的特征向量来构建文章特征，我们也迅速实现了这样的结构。但手动测试发现结果不尽如人意，其中推荐出来的文章约有一半都关系不大。在我们仔细思考 TF-IDF 的原理后我们发现 TF-IDF 并没有考虑文本语义，例如“choose”和“choosing”实际上可以当做一个词但在 TF-IDF 中是当做两个词两组频率来计算的，理论上只要运气好，两个毫不相关的文章也可能出现 TF-IDF 近似的情况，这就是由于没有考虑语义而可能出现的极端情况。为此，我们开始考虑添加语义的方法，但我们并没有彻底放弃用 TF-IDF，我们知道，TF-IDF 具有能够提取出文章关键词的优良特性，其计算公式如下：

$$\mathrm{tf}(w,d)=\frac{f(w,d)}{\sum_i f(w_i,d)}$$

$$idf_w=\log\left(\frac{N}{1+df_w}\right)$$

其中，$f(w,d)$表示单词 w 在文档 d 中出现的次数，df_w 表示语料库中包含词 w 的文档数量，N 表示语料库中的全部文档数量。

词条 w 的权重为：$\mathrm{tf\_id}f_w=\mathrm{tf}(w,d)\times idf_w$

对于一个几乎在每个文档中都出现的单词 w，它十分常见，不论它是名字还是动词，它都不能体现文档的特征，故此时 idfw 几乎为 0，从而 tf-idf 也几乎为 0，该词 tf-idf 频率很低。

对于一个只在几个文档中出现过的词，它就是我们感兴趣的可以代表这几个文档特征的词，从而这个时候的 idfw 就比较大，所以 tf-idfw 也比较大，该词的 tf-idf 频率较大。

由上分析我们可知通过对每篇文章所有词的 TF-IDF 排序我们可以选出每篇文章的 N 个重要特征词，在这里我们把 N 设为了 12。根据 TF-IDF 的原理我们认定这 N 个特征词代表了这篇文章。接下来就是如何通过这些关键词来构建考虑语义的论文特征了。

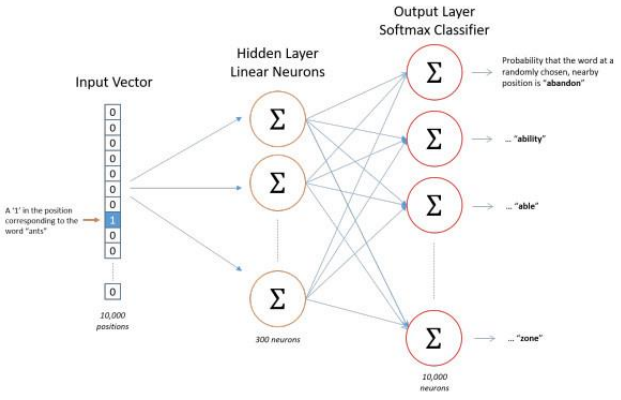
二、深度学习计算词向量

Word2Vec 模型介绍：

在这次实验中我们组使用了 Word2Vec 模型，它是一种从大量文本语料中以无监督的方式学习语义知识的一种模型，它被大量地用在 NLP 中。Word2Vec 其实就是通过学习文本来用词向量的方式表征词的语义信息，即通过一个嵌入空间使得语义上相似的单词在该空间内距离很近。这个模型考虑了语义，结合我们之前提取的特征词，经过一定的训练之后我们就可以构建每个文档的词向量。

我们从直观角度上来理解一下，cat 这个单词和 kitten 属于语义上很相近的词，而 dog 和 kitten 则不是那么相近，iphone 这个单词和 kitten 的语义就差的更远了。通过对词汇表中单词进行这种数值表示方式的学习（也就是将单词转换为词向量），能够让我们基于这样的数值进行向量化的操作从而得到一些有趣的结论。比如说，如果我们对词向量 kitten、cat 以及 dog 执行这样的操作：kitten - cat + dog，那么最终得到的嵌入词向量将与 puppy 这个词向量十分相近。

Word2Vec 模型中，主要有 Skip-Gram 和 CBOW 两种模型，从直观上理解，Skip-Gram 是给定 input word 来预测上下文。而 CBOW 是给定上下文，来预测 input word。由于我们仅使用了 Skip-Gram 模型，因此以下只介绍 Skip-Gram 模型。



Skip-Gram 模型实际上分为两个部分，第一部分为建立模型，第二部分是通过模型获取嵌入词向量。实际上这个模型

是基于无监督学习的，整个建模过程和自编码机十分相似。即先基于训练数据构建一个神经网络，当这个模型训练好以后，我们并不会用这个训练好的模型处理新的任务，我们真正需要的是这个模型通过训练数据所学得的参数，即隐层的权重矩阵——后面我们将会看到这些权重在 Skip-Gram 模型中实际上就是我们试图去学习的词向量。

训练：

我们训练这个模型的真正目的是获得模型基于训练数据学得的隐层权重。但为了获得这些权重，我们必须首先要构建并训练一个完整的神经网络。

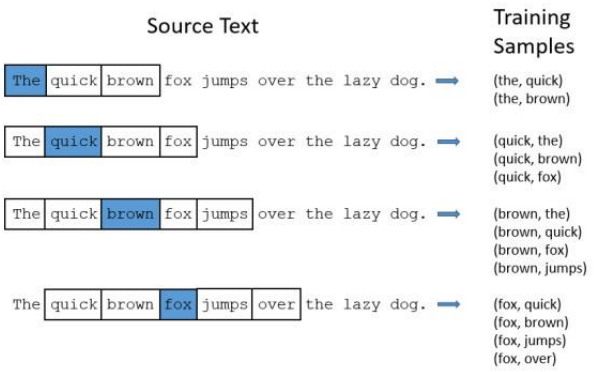
接下来说说我们如何训练我们的神经网络。假如我们有一个句子“The dog barked at the mailman”。

① 首先我们选句子中间的一个词作为我们的输入词，例如我们选取“dog”作为 input word；

② 有了 input word 以后，我们再定义一个叫做 skip_window 的参数，它代表着我们从当前 input word 的一侧（左边或右边）选取词的数量。如果我们设置 skip_window=2，那么我们最终获得窗口中的词（包括 input word 在内）就是 ['The', 'dog', 'barked', 'at'] 。skip_window=2 代表着选取 input word 左侧的 2 个词和右侧的 2 个词进入我们的窗口，所以整个窗口大小 span=2x2=4。另一个参数叫 num_skips，它代表着我们从整个窗口中选取多少个不同的词作为我们的 output word，当 skip_window=2，num_skips=2 时，我们将会得到两组 (input word, output word) 形式的训练数据，即 ('dog', 'barked')，('dog', 'the')。

③ 神经网络基于这些训练数据将会输出一个概率分布，这个概率代表着我们的词典中的每个词是 output word 的可能性。比如第二步中我们在设置 skip_window 和 num_skips=2 的情况下获得了两组训练数据。假如我们先拿一组数据 ('dog', 'barked') 来训练神经网络，那么模型通过学习这个训练样本，会告诉我们词汇表中每个单词是“barked”的概率大小。

模型的输出概率代表着到我们词典中每个词有多大可能性跟 input word 同时出现。举个例子，如果我们向神经网络模型中输入一个单词“Soviet”，那么最终模型的输出概率中，像“Union”，“Russia”这种相关词的概率将远高于像“watermelon”，“kangaroo”非相关词的概率。因为“Union”，“Russia”在文本中更大可能在“Soviet”的窗口中出现。我们将通过给神经网络输入文本中成对的单词来训练它完成上面所说的概率计算。下面的图中给出了一些我们的训练样本的例子。我们选定句子“The quick brown fox jumps over lazy dog”，设定我们的窗口大小为 2（window_size=2），也就是说我们仅选输入词前后各两个词和输入词进行组合。下图中，蓝色代表 input word，方框内代表位于窗口内的单词。



我们的模型将会从每对单词出现的次数中习得统计结果。例如，我们的神经网络可能会得到更多类似（“Soviet”，“Union”）这样的训练样本对，而对于（“Soviet”，“Sasquatch”）这样的组合却看到的很少。因此，当我们的模型完成训练后，给定一个单词“Soviet”作为输入，输出的结果中“Union”或者“Russia”要比“Sasquatch”被赋予更高的概率。

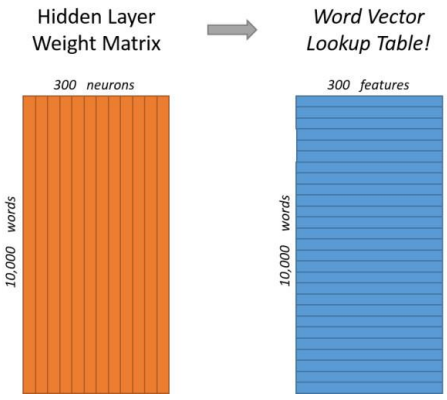
隐层：

最后再来介绍下隐层。如果我们现在有 10000 个单词，并且我们现在想用 300 个特征来表示一个单词（即每个词可以被表示为 300 维的向量）。出于效率原因，我们对 input word 和 output word 都进行了 one-hot 编码。这样隐层的权重矩阵应该为 10000 行，300 列（隐层有 300 个结点）。

Google 在最新发布的基于 Google news 数据集训练的模型中使用的就是 300 个特征的词向量。词向量的维度是一个可以调节的超参数（在 Python 的 gensim 包中封装的 Word2Vec 接口默认的词向量大小为 100， window_size 为 5）。

看下面的图片，左右两张图分别从不同角度代表了输入层-隐层的权重矩阵。左图中每一列代表一个 10000 维的词向量和隐层单个神经元连接的权重向量。从右边的图来看，每一行实际上代表了每个单词的词向量。

注：隐层没有使用任何激活函数



所以我们最终的目的就是学习这个隐层的权重矩阵。由于篇幅有限，具体细节此处不再介绍。

综上，我们可以通过这个模型构建单词的词向量，然后将每篇文章的词向量累加作为每篇文章的综合特征。敲定了这个想法以后，我们组首先找来了 Google 前几年发布的基于几亿个新闻语料单词训练的三百万个单词的三百维词向量数据看看效果，果然在实际提取中发现，这个数据库并不适合我们的项目，因为我们的项目是基于学术论文英语语料的，很多专有名词在这个数据库中都没有。于是，我们便尝试自己找数据训练，但是最终没有找到比较好的免费学术论文英语语料数据集。于是我们便把 raw-data.txt 里的所有标点符号和停用词都去掉，然后把这个共有两百万单词的语料库作为训练集进行训练，训练结束后对于每个文章的 N 个特征词，我们将它的特征词的词向量向量相加来作为这篇文章的特征向量。

注：训练 Skip-Gram 模型利用了 python 的自然语言处理包 gensim。

三、推荐文章：

既然上一步我们构建了每篇文章的特征向量，接下来我们就可以通过这些向量来给用户推荐文章了。

那么应该如何对用户推荐文章呢？查阅资料之后我们最终找到了两种方法：

一：将用户读过所有文章的词向量取平均作为其用户的特征向量，在测试集中对每个论文与用户特征向量求余弦距离，余弦值越大我们认为其越相关，然后排序取余弦值最大的 5 个推荐即可。

二：不对用户的特征向量建模，在测试集中对某个用户的某个测试论文求它与该用户读过所有论文的余弦值最大的那个，即最相近论文之间的余弦距离作为该测试论文与该用户的距离，然后排序取余弦值最大的 5 个推荐即可。

进一步讨论之后我们选取了前一种方法。因为每个用户读过的论文的特征向量略有不同，虽然差距不大但都有一定的差距。我们不能认为与其中一篇论文十分相近就认为该用户会喜欢这些论文。我们认为即使用户读过某些论文，我们也不能一视同仁，用户对于不同论文的喜爱程度是有差别的。但此次实验的结果未知，我们所做的工作类似于无监督学习。所以我们没办法调整用户所读论文词向量的权重来表示用户对于所读论文的喜爱程度。因此，第二种方法是不可行的，于是我们便用了第一种方法，对所有用户读过的文章取平均词向量作为其特征向量。我们也测试了一下，我们发现如果我们把 train 集作为 test 集，即计算将所读论文的平均向量作为用户的特征向量之后对用户求其已经读过的论文与用户的关系，我们发现大部分余弦值都在 0.8 以上，极少数在 0.6-0.8 附近，这表明用户读过的论文大部分都和用户喜爱看的论文相关，极少部分用户看了也觉得“一般”。然后我们又用了 test 集测试，发现大部分文章余弦值都在 0.2-0.5 附近，极少部分能到 0.7 以上，这也符合预期。我们手动检验了这些论文，发现近似是相关的。

由此，我们的实验结束。

不足之处：

① 由于没有找到比较好的数据集，所以我们训练的数据集只是 200 万个单词，效果应该不是很好，如果能找到亿级的基于学术论文语料的数据集，相信会有一个更好的表现。

② 此次实验我们的测试做的不好，没有提出有效的测试方案，大多数测试都是手动观看，这耗费了大量的时间。

③ 实际上在提取每篇文章的特征词的时候我们应该考虑一下短语，比如把“data structure”分为“data”和“structure”就会出现误差，后来由于没有测试方案，也不知道这样子能不能提升效果，能的话能提升多少，于是就此作罢。

细致流程：

① 对于每篇文章，去掉所有的标点符号。

② 对于每篇文章，去掉停用词。(即”and”,”am”,”do”等对文章内容并无实际帮助的词)

③ 对每篇文章的所有词进行 TF-IDF 计算，将 TF-IDF 最大的 12 个词（若不足 12 个则有几个存储几个）存储。

④ 训练 skip-gram 模型。

⑤ 利用 skip-gram 模型的隐层矩阵构建每篇文章的特征向量。

⑥ 利用用户所读论文的特征向量给用户建模。

⑦ 在 test 集中计算测试论文的余弦距离，将最相关的 5 个返回。

⑧ 程序结束

参考文献：

[1] Ramos, Juan Pramyr. “Using TF-IDF to Determine Word Relevance in Document Queries.” (2003).

[2] 邬明强,张奎.结合 TFIDF 方法与 Skip-gram 模型的文本分类方法研究[J].电子技术与软件工程,2018(06):162-163.

[3] 张剑,屈丹,李真.基于词向量特征的循环神经网络语言模型[J].模式识别与人工智能,2015,28(04):299-305.

[4]王妍,唐杰.基于深度学习的论文个性化推荐算法[J].中文信息学报,2018,32(04):114-119.32(04):114-119.

[5]唐明,朱磊,邹显春.基于 Word2Vec 的一种文档向量表示[J].计算机科学,2016,43(06):214-217+269.