一、问题描述

利用给定语料库（或者自选语料库），利用神经语言模型（如：Word2Vec，GloVe 等模型）来训练词向量，通过对词向量的聚类或者其他方法来验证词向量的有效性。

二、实验原理

1. Word Embedding

1.1 基本概念

词向量（Word Embedding），又叫 Word 嵌入式自然语言处理（NLP）中的一组语言建模和特征学习技术的统称，其中来自词汇表的单词或短语被映射到实数的向量。从概念上讲，它涉及从每个单词一维的空间到具有更低维度的连续向量空间的数学嵌入。

如果将 word 看作文本的最小单元，可以将 Word Embedding 理解为一种映射，其过程是：将文本空间中的某个 word，通过一定的方法，映射或者说嵌入（Embedding）到另一个数值向量空间（之所以称之为 Embedding，是因为这种表示方法往往伴随着一种降维的意思）。

生成这种映射的方法包括神经网络，单词共生矩阵的降维，概率模型，可解释的知识库方法，和术语的显式表示单词出现的背景。

当用作底层输入表示时，单词和短语嵌入已经被证明可以提高 NLP 任务的性能，例如语法分析和情感分析。

1.2 输入和输出

Word Embedding 的输入是原始文本中的一组不重叠的词汇，假设有句子：what is your name?那么为了便于处理，我们可以将这些词汇放置到一个 dictionary 里，例如：[“what”, “is”, “your”, “name”]，这个 dictionary 就可以看作是 Word Embedding 的一个输入。

Word Embedding 的输出就是每个 word 的向量表示。对于上文中的原始输入，假设使用最简单的 one-hot 编码方式，那么每个 word 都对应了一种数值表示。例如，what 对应的 vector 就是[1,0,0,0]，your 对应的 vector 就是[0,0,1,0]，各种机器学习应用可以基于这种 word 的数值表示来构建各自的模型。当然，这是一种最简单的映射方法，但却足以阐述 Word Embedding 的意义。下文将介绍常

见的 Word Embedding 的方法和优缺点。

1.3 Word Embedding 的类型

Word Embedding 也是有流派的，主流有以下两种：

1.3.1 基于频率的 Word Embedding（Frequency Based Embedding）

主要有以下几种：

(1) Count Vector

(2) TF-IDF Vector

(3) Co-Occurrence Vector

1.3.2 基于预测的 Word Embedding（Prediction Based Embedding）

主流的有以下两种：

(1) CBOW（Continues Bag Of Words）

(2) Skip–Gram

1.4 Word Embedding 的应用

现今流行的 Word Embedding 算法携带了语义信息且维度经过压缩便于运算，因此有很多用途，例如：

(1) 计算相似度

(2) 在一组单词中找出与众不同的一个，例如在如下词汇列表中：[dog, cat, chicken, boy]，利用词向量可以识别出 boy 和其他三个词不是一类。

(3) 直接进行词的运算，例如经典的：woman + king – man = queen

(4) 由于携带了语义信息，还可以计算一段文字出现的可能性，也就是说，这段文字是否通顺。

本质上来说，经过 Word Embedding 之后，各个 word 就组合成了一个相对低维空间上的一组向量，这些向量之间的远近关系则由他们之间的语义关系决定。

2. Word2vec 模型

Word2vec，为一群用来产生词向量的相关模型。这些模型为浅层双层的神经网络，用来训练以重新建构语言学之词文本。网络以词表现，并且需猜测相邻位置的输入词，在 word2vec 中词袋模型假设下，词的顺序是不重要的。

训练完成之后，word2vec 模型可用来映射每个词到一个向量，可用来表示词对词之间的关系。该向量为神经网络之隐藏层。

Word2vec 依赖 skip-grams 或 CBOW（连续词袋）来建立神经词嵌入。

2.1 Skip-gram 和 CBOW 的简单情形

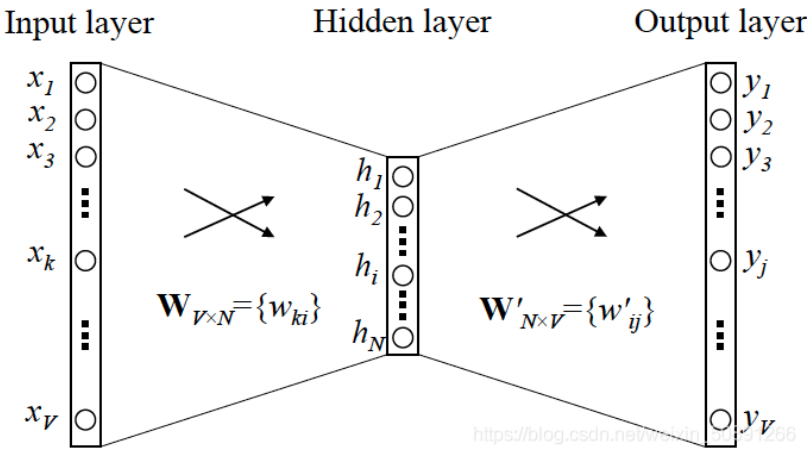
2.1.1 one-hot encoder

one-hot encoder，其思想跟特征工程里处理类别变量的 one-hot 一样，本质上是用一个只含一个 1、其他都是 0 的向量来唯一表示词语。

举个例子，假设全世界所有的词语总共有 V 个，这 V 个词语有自己的先后顺序，假设“我”这个词是第 1 个词，“你”这个单词是第 2 个词，那么“我”就可以表示为一个 V 维全零向量、把第 1 个位置的 0 变成 1，而“你”同样表示为 V 维全零向量、把第 2 个位置的 0 变成 1。这样，每个词语都可以找到属于自己的唯一表示。

2.1.2 简单情形下的 Skip-gram 和 CBOW

接下来来看 Skip-gram 的网络结构了，x 就是上面提到的 one-hot encoder 形式的输入，y 是在这 V 个词上输出的概率，我们希望跟真实的 y 的 one-hot encoder 一样。



在上述图示中，隐层的激活函数其实是线性的，相当于没做任何处理，我们要训练这个神经网络，用反向传播算法，本质上是链式求导。

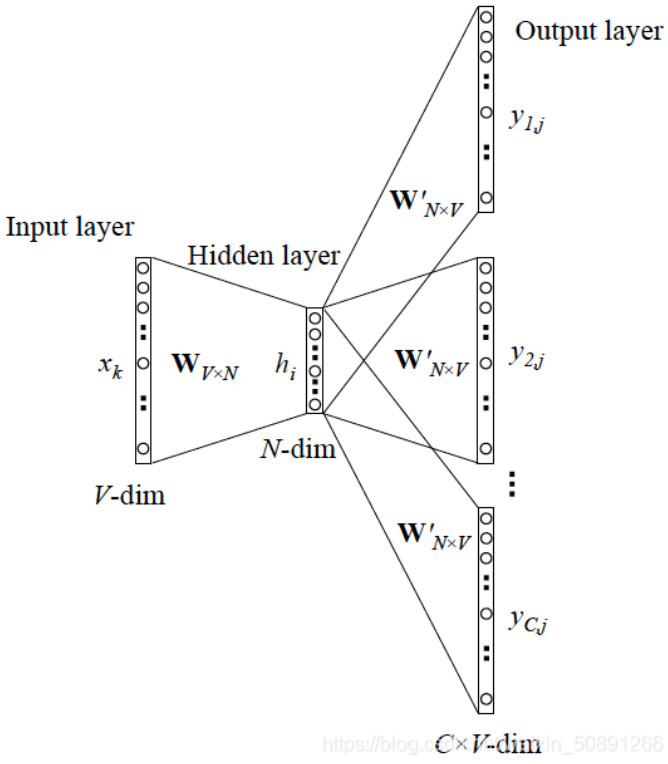
当模型训练完后，最后得到的其实是神经网络的权重，比如现在输入一个x的 one-hot encoder：[1,0,0,…,0]，对应刚说的那个词语“我”，则在输入层到隐含层的权重里，只有对应 1 这个位置的权重被激活，这些权重的个数，跟隐含层节点数是一致的，从而这些权重组成一个向量$\overrightarrow{v_x}$来表示x，而因为每个词语的 one-hot encoder 里面 1 的位置是不同的，所以，这个向量$\overrightarrow{v_x}$就可以用来唯一表示x。

此外，输出y也是用V个节点表示的，对应V个词语，所以把输出节点置成[1,0,0,…,0]，它也能表示“我”这个单词，但是激活的是隐含层到输出层的权重，这些权重的个数，跟隐含层一样，也可以组成一个向量$\overrightarrow{v_y}$，跟上面提到的$\overrightarrow{v_x}$维度一样，并且可以看做是词语“我”的另一种词向量。而这两种词向量$\overrightarrow{v_x}$和$\overrightarrow{v_y}$，正是Mikolov 在论文里所提到的，“输入向量”和“输出向量”，一般我们用“输入向量”。

这个词向量的维度（与隐含层节点数一致）一般情况下要远远小于词语总数 V 的大小，所以 Word2vec 本质上是一种降维操作——把词语从 one-hot encoder 形式的表示降维到 Word2vec 形式的表示。

2.2 Skip-gram

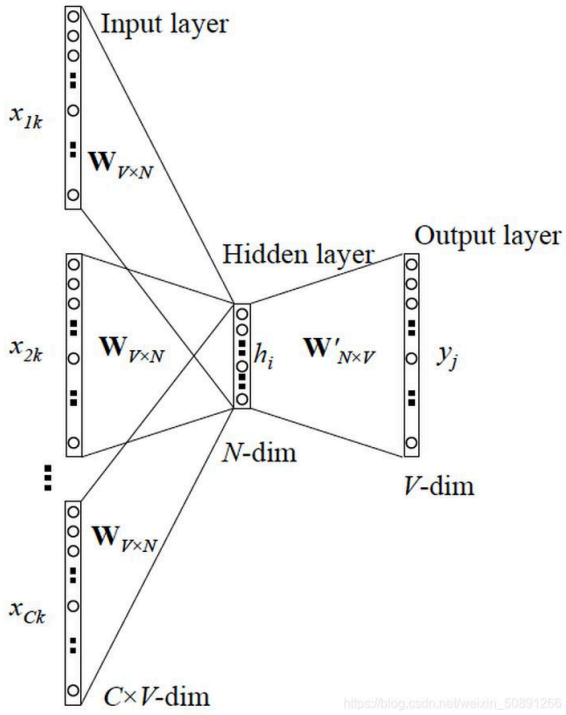
上面讨论的是 Skip-gram 的最简单情形，即y只有一个词，当y有多个词时，网络结构如下：



可以看成是单个x到单个y模型的并联，cost function 是单个 cost function 的累加（取log之后）。

2.3 CBOW

CBOW 跟 Skip-gram 相似，Skip-gram 是预测一个词的上下文，而 CBOW 是用上下文预测这个词，网络结构如下：



与 Skip-gram 的模型并联不同，这里是输入变成了多个单词，所以要对输入处理下（一般是求和然后平均），输出的 cost function 不变。

三、实验过程与结果分析

1. 实验过程

由于对金庸小说不熟悉，所以选择了《三体》作为本次实验的语料库。过滤包含在“stop_words.txt”中的停词，并在 jieba 分词的基础上补充“三体词典.txt”中的词汇。使用 jieba 分词并调用 gensim 包中的 Word2Vec 函数，进行词向量的训练。训练采用 Skip-gram 模型。最后通过模型中的 similarity、most_similar 等函数对各个要素进行分析。

2. 实验结果与分析

2.1 与给定人物最相关的前 10 个词

汪淼		罗辑		程心	
词语	相关程度	词语	相关程度	词语	相关程度
大史	0.919187725	史强	0.945620656	AA	0.950969994
史强	0.885283709	大史	0.919909596	维德	0.870810807
发现	0.86932832	庄颜	0.919081271	很快	0.844773769
明白	0.860215843	孩子	0.853956044	本来	0.834724784
眼睛	0.855686426	真的	0.850570023	打开	0.832297444
很快	0.846116304	汪淼	0.843076468	留下	0.823503137
罗辑	0.843076587	身上	0.834952176	明白	0.816598475
东西	0.842975855	云天明	0.833204746	阳光	0.813491583
简单	0.839550316	告诉	0.831403792	身上	0.812042236
庄颜	0.835098088	回答	0.831046402	画面	0.806148171
章北海		大史		史强	
词语	相关程度	词语	相关程度	词语	相关程度
舰长	0.951364934	史强	0.985704899	大史	0.985704899
军官	0.941343427	庄颜	0.929985762	罗辑	0.945620716
东方	0.929327786	罗辑	0.919909596	庄颜	0.935383856
一名	0.856724501	汪淼	0.919187725	身上	0.886053503
自然选择	0.837001383	眼睛	0.889993072	汪淼	0.885283709
丁仪	0.820571721	打开	0.886568964	打开	0.884114742
关一帆	0.792296588	回答	0.883668602	目光	0.879213929
本来	0.781456709	东西	0.877684772	回答	0.878620684
维德	0.764418602	身上	0.876960754	明白	0.869795978
工作	0.761944473	目光	0.871407807	孩子	0.865946591

上表中数据基本符合《三体》小说中人物关系：汪淼、罗辑二人接触最多的人物就是史强（大史），史强也帮罗辑找到了梦中情人庄颜；艾 AA（文中多称为 AA）是程心与联合国的联络人，维德跟程心也关系复杂；章北海和东方延绪（文中多称为东方）随自然选择号飞船一同逃亡。以及，程心心里根本没有云天明。

有趣的是，“史强”和“大史”的相关度高达 0.985704899（直逼韦德的威慑度），远远高于其他词语之间的相关程度。如果将这种相关性认为表示词义相同的话，那么可以说，史强作为书中唯一一个有两个被频繁使用的称呼的人，不负众望地帮助机器证明了神经网络是读得懂小说的。

2.2 两个词语之间的相关程度

计算部分不在表中的人物之间的相关程度，得到：

	汪淼	罗辑
叶文洁	0.623711944	0.919081211
云天明	0.696082115	0.833204687
章北海	0.463029742	0.380149305

在我看来，两位主角与叶文洁、云天明、章北海三人的相关程度均依次递减，结果基本能证实这一看法。但令人意外的是：汪淼与叶文洁的相关程度仅有 0.623711944。阅读原文后得出原因：两人对话的情节中，汪淼对叶文洁的称呼是“叶老师”，而后者被作者称为“杨母”，所以即使两人有联系，但“汪淼”与“叶文洁”两词在词向量模型中相关性很低。

2.3 最小词频对实验的影响

叶文洁		叶文洁	
词语	相关程度	词语	相关程度
红岸	0. 886488259	杨卫宁	0. 883488238
基地	0. 867664099	雷志成	0. 866579771
想到	0. 781489849	红岸	0. 805740237
父亲	0. 77179122	政委	0. 801749349
工作	0. 771755755	基地	0. 780652583
明白	0. 756894529	谈话	0. 772930145
告诉	0. 747299612	文件	0. 767979503
回答	0. 736086547	明白	0. 754487634
肯定	0. 733551502	没什么	0. 754370332
地说	0. 730232239	回答	0. 753598452

对于叶文洁，由于首次试验中，生成模型时过滤了词频小于 100 的词，丈夫杨卫宁与政委雷志成两个与叶文洁联系密切，但出场仅数十次的人名被剔除了。修改 min_count 参数重新训练后，得到如上表右侧结果，可以看出结果较为准确。

遗憾的是，由于《三体》相比金庸小说，重故事轻人物，人物情感并不丰富，也没有十分亲密的人物关系，很难找出像老师课上说的郭靖-黄蓉=张无忌 - 赵敏之类的结果。

四、参考资料

[1] https://blog.csdn.net/weixin_50891266/article/details/116750204

[2] https://zhuanlan.zhihu.com/p/490316695