词向量工具word2vec的学习

人工智能与大数据技术 2017-02-20

> **来自：标点符的《词向量工具word2vec的学习》**
> **链接：https://www.biaodianfu.com/word2vec.html（点击尾部阅读原文前往）**

什么是word2vec?

word2vec是Google在2013年开源的一款将词表征为实数值向量（word vector）的高效工具，采用的模型有CBOW（Continuous Bag-Of-Words，即连续的词袋模型）和Skip-Gram两种。

word2vec通过训练，可以把对文本内容的处理简化为K维向量空间中的向量运算，而向量空间上的相似度可以用来表示文本语义上的相似度。因此，word2vec输出的词向量可以被用来做很多NLP相关的工作，比如聚类、找同义词、词性分析等等。Word2vec的应用不止于解析自然语句。它还可以用于基因组、代码、点赞、播放列表、社交媒体图像等其他语言或符号序列，同样能够有效识别其中存在的模式。为什么呢？因为这些数据都是与词语相似的离散状态，而我们的目的只是求取这些状态之间的转移概率，即它们共同出现的可能性。所以gene2vec、like2vec和follower2vec都是可行的。

而word2vec被人广为传颂的地方是其向量的加法组合运算（Additive Compositionality），官网上的例子是：vector('Paris') -vector('France') + vector('Italy')≈vector('Rome')，vector('king') -vector('man') + vector('woman')≈ vector('queen')。但这个多少有点被过度炒作了，很多其他降维或主题模型在一定程度也能达到类似效果，而且word2vec也只是少量的例子完美符合这种加减法操作，并不是所有的case都满足。

word2vec一般被外界认为是一个Deep Learning（深度学习）的模型，究其原因，可能和word2vec的作者Tomas Mikolov的Deep Learning背景以及word2vec是一种神经网络模型相关，但我们谨慎认为该模型层次较浅，严格来说还不能算是深层模型。当然如果word2vec上层再套一层与具体应用相关的输出层，比如Softmax，此时更像是一个深层模型。Word2vec是一个用于处理文本的双层神经网络。它的输入是文本语料，输出则是一组向量：该语料中词语的特征向量。虽然Word2vec并不是深度神经网络，但它可以将文本转换为深度神经网络能够理解的数值形式。Word2vec衡量词的余弦相似性，无相似性表示为90度角，而相似度为1的完全相似则表示为0度角，即完全重合。

word2vec是一个将单词转换成向量形式的工具。可以把对文本内容的处理简化为向量空间中的向量运算，计算出向量空间上的相似度，来表示文本语义上的相似度。word2vec大受欢迎的另一个原因是其高效性，Mikolov在论文中指出一个优化的单机版本一天可训练上千亿词。

什么是词向量?

自然语言理解的问题要转化为机器学习的问题，第一步肯定是要找一种方法把这些符号数学化。词向量就是用来将语言中的词进行数学化的一种方式，顾名思义，词向量就是把一个词表示成一个向量。主要有两种表示方式：
One-Hot Representation

NLP相关任务中最常见的第一步是创建一个词表库并把每个词顺序编号。这实际就是词表示方法中的One-hot Representation，这种方法把每个词顺序编号，每个词就是一个很长的向量，向量的维度等于词表大小，只有对应位置上的数字为1，其他都为0。当然在实际应用中，一般采用稀疏编码存储，主要采用词的编号。举个例子：

- “话筒”表示为 [0 0 0 1 00 0 0 0 0 0 0 0 0 0 0 ...]
- “麦克”表示为 [0 0 0 0 00 0 0 1 0 0 0 0 0 0 0 ...]

这种 One-hotRepresentation 如果采用稀疏方式存储，会是非常的简洁：也就是给每个词分配一个数字 ID。比如刚才的例子中，话筒记为 3，麦克记为 8（假设从 0 开始记）。如果要编程实现的话，用 Hash 表给每个词分配一个编号就可以了。这么简洁的表示方法配合上最大熵、SVM、CRF 等等算法已经很好地完成了 NLP 领域的各种主流任务。这种表示方法一个最大的问题是无法捕捉词与词之间的相似度，就算是近义词也无法从词向量中看出任何关系。此外这种表示方法还容易发生维数灾难，尤其是在Deep Learning相关的一些应用中。
Distributed Representation

Distributed representation 最早是 Hinton 在 1986 年的论文《Learning distributed representations of concepts》中提出的。其基本思想是通过训练将每个词映射成K维实数向量（K一般为模型中的超参数），通过词之间的距离（比如cosine相似度、欧氏距离等）来判断它们之间的语义相似度。虽然这篇文章没有说要将词做Distributed representation，但至少这种先进的思想在那个时候就在人们的心中埋下了火种，到 2000 年之后开始逐渐被人重视。
Distributed representation 用来表示词，通常被称为“Word Representation”或“Word Embedding”，中文俗称“词向量”。

Distributed representation 最大的贡献就是让相关或者相似的词，在距离上更接近了。向量的距离可以用最传统的欧氏距离来衡量，也可以用余弦夹角来衡量。用这种方式表示的向量，“麦克”和“话筒”的距离会远远小于“麦克”和“天气”。可能理想情况下“麦克”和“话筒”的表示应该是完全一样的，但是由于有些人会把英文名“迈克”也写成“麦克”，导致“麦克”一词带上了一些人名的语义，因此不会和“话筒”完全一致。word2vec使用的就是这种Distributed representation的词向量表示方式。

word2vec的模型训练

Word2vec与自动编码器相似，它将每个词编码为向量，但Word2vec不会像受限玻尔兹曼机那样通过重构输入的词语来定型，而是根据输入语料中相邻的其他词来进行每个词的定型。具体的方式有两种，一种是用上下文预测目标词（连续词袋法，简称CBOW），另一种则是用一个词来预测一段目标上下文，称为skip-gram方法。这两种方法都利用人工神经网络作为它们的分类算法。

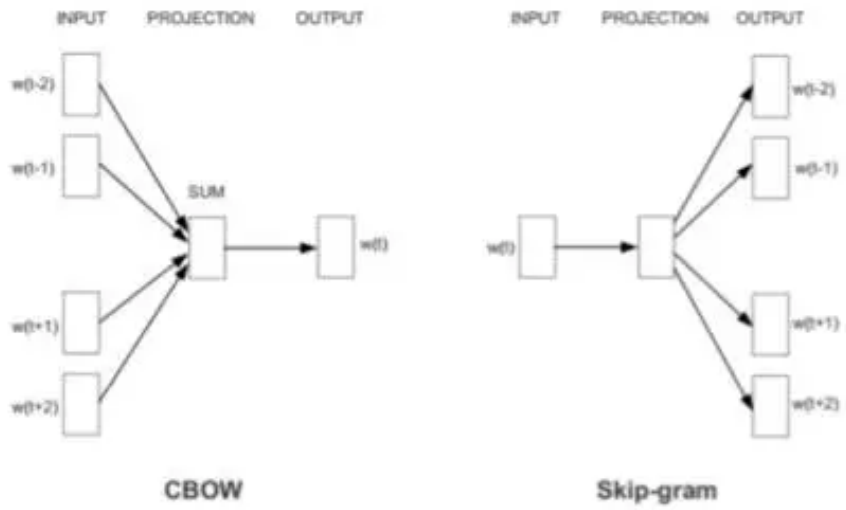

Figure 1: Architecture for the CBOW and Skip-gram method, taken from Efficient Estimation of Word Representations in Vector Space. W(t) is the current word, while w(t-2), w(t-1), etc. are the surrounding words.

- CBOW是Continuous Bag-of-Words Model的缩写，是一种与前向NNLM类似的模型，不同点在于CBOW去掉了最耗时的非线性隐层且所有词共享隐层。如上图所示。可以看出，CBOW模型是预测P(wt|wt-k,wt-(k-1)...,wt-1,wt+1,wt+2...,wt+k)。
- Skip-Gram模型的图与CBOW正好方向相反，从图中看应该Skip-Gram应该预测概率p(wi|wt)，其中−≤≤+且≠，c是决定上下文窗口大小的常数，c越大则需要考虑的pair就越多，一般能够带来更精确的结果，但是训练时间也会增加。

开源代码：

除了google自己的word2vec工具，各位对词向量感兴趣的牛人们也相继编写了各自不同的版本。其中比较好用的是Python Gensim主题模型包中的word2vec，通过阅读其源码python版本只实现了skip-gram模型，并且只实现了通过分层softmax方法对其训练，并没有使用negative sampling。

- **C语言：** https://github.com/dav/word2vec（在原版的基础上打了一些社区的patch）
- **Python：** https://github.com/RaRe-Technologies/gensim

参考资料：

- http://techblog.youdao.com/?p=915
- https://www.douban.com/note/298095260/
- http://licstar.net/archives/328
- https://www.zhihu.com/question/21661274
- http://blog.csdn.net/mytestmy/article/details/26961315
- http://blog.csdn.net/mytestmy/article/details/26969149
- http://www.cnblogs.com/peghoty/p/3857839.html
- https://deeplearning4j.org/cn/archieved/zh-word2vec
- http://www.hankcs.com/nlp/word2vec.html
- http://aial.shiroyagi.co.jp/2015/12/word2vec/
- http://blog.sina.com.cn/s/blog_a89e19440101oxvr.html
- https://www.zybuluo.com/Dounm/note/591752
