个人资料

皮果提

+ 加关注 发私信

访问： 1224446次
积分： 19134
等级： BLOG 7
排名： 第454名

原创： 104篇
转载： 13篇
译文： 2篇
评论： 750条

文章分类

数据挖掘 (35)
深度学习 (20)
语言模型 (10)
文本挖掘 (1)
强化学习 (1)
数学天地 (18)
编程知识 (5)
隐马模型 (8)
杂七杂八 (14)
机器学习 (37)
并行计算 (3)

阅读排行

word2vec 中的数学原理i (93579)
word2vec 中的数学原理i (55150)
受限玻尔兹曼机（RBM） (42965)
word2vec 中的数学原理i (39988)
word2vec 中的数学原理i (39890)
word2vec 中的数学原理i (39408)
牛顿法与拟牛顿法学习笔 (33615)
牛顿法与拟牛顿法学习笔 (32578)
牛顿法与拟牛顿法学习笔 (32361)
受限玻尔兹曼机（RBM） (31081)

评论排行

word2vec 中的数学原理i (153)
word2vec 中的数学原理i (61)
Community Detection 算 (60)

原 word2vec 中的数学原理详解（三）背景知识

标签： word2vec CBOW Skip-gram Hierarchical Softmax Negative Sampling

2014-07-19 22:49 39898人阅读 评论(43) 收藏 举报

分类： 语言模型（9）

word2vec 是 Google 于 2013 年开源推出的一个用于获取 word vector 的工具包，它简单、高效，因此引起了很多人的关注。由于 word2vec 的作者 Tomas Mikolov 在两篇相关的论文 [3,4] 中并没有谈及太多算法细节，因而在一定程度上增加了这个工具包的神秘感。一些按捺不住的人于是选择了通过解剖源代码的方式来一窥究竟，出于好奇，我也成为了他们中的一员。读完代码后，觉得收获颇多，整理成文，给有需要的朋友参考。

相关链接

（一）目录和前言
（二）预备知识
（三）背景知识
（四）基于 Hierarchical Softmax 的模型
（五）基于 Negative Sampling 的模型
（六）若干源码细节

受限玻尔兹曼机（RBM） (48)
word2vec 中的数学原理 (43)
发表在 Science 上的一种 (32)
受限玻尔兹曼机（RBM） (26)
利用 word2vec 训练的字 (22)
受限玻尔兹曼机（RBM） (21)
word2vec 中的数学原理 (20)

§3 背景知识

word2vec 是用来生成词向量的工具，而词向量与语言模型有着密切的关系，为此，不妨先来了解一些语言模型方面的知识.

§3.1 统计语言模型

当今的互联网迅猛发展，每天都在产生大量的文本、图片、语音和视频数据，要对这些数据进行处理并从中挖掘出有价值的信息，离不开自然语言处理 (Nature Language Processing, NLP) 技术，其中**统计语言模型** (Statistical Language Model) 就是很重要的一环，它是所有 NLP 的基础，被广泛应用于语音识别、机器翻译、分词、词性标注和信息检索等任务.

例 3.1 *在语音识别系统中，对于给定的语音段 $Voice$，需要找到一个使概率 $p(Text|Voice)$ 最大的文本段 $Text$. 利用 Bayes 公式，有*

$$p(Text|Voice) = \frac{p(Voice|Text) \cdot p(Text)}{p(Voice)},$$

*其中 $p(Voice|Text)$ 为**声学模型**，而 $p(Text)$ 为**语言模型** ([18]).*

简单地说，统计语言模型是用来计算一个句子的概率的**概率模型**，它通常基于一个语料库来构建. 那什么叫做一个句子的概率呢? 假设 $W = w_1^T := (w_1, w_2, \cdots, w_T)$ 表示由 T 个词 $w_1, w_2, \cdots, w_T$ 按顺序构成的一个句子，则 $w_1, w_2, \cdots, w_T$ 的联合概率

$$p(W) = p(w_1^T) = p(w_1, w_2, \cdots, w_T)$$

就是这个句子的概率. 利用 **Bayes 公式**，上式可以被链式地分解为

$$p(w_1^T) = p(w_1) \cdot p(w_2|w_1) \cdot p(w_3|w_1^2) \cdots p(w_T|w_1^{T-1}), \tag{3.1}$$

其中的 (条件) 概率 $p(w_1)$, $p(w_2|w_1)$, $p(w_3|w_1^2)$, $\cdots$, $p(w_T|w_1^{T-1})$ 就是**语言模型的参数**，若这些参数已经全部算得，那么给定一个句子 w_1^T，就可以很快地算出相应的 $p(w_1^T)$ 了.

看起来好像很简单，是吧? 但是，具体实现起来还是有点麻烦. 例如，先来看看**模型参数的个数**. 刚才是考虑一个给定的长度为 T 的句子，就需要计算 T 个参数. 不妨假设语料库对应词典 $\mathcal{D}$ 的大小 (即词汇量) 为 N，那么，如果考虑长度为 T 的任意句子，理论上就有 N^T 种可能，而每种可能都要计算 T 个参数，总共就需要计算 TN^T 个参数. 当然，这里只是简单估算，并没有考虑重复参数，但这个量级还是有蛮吓人. 此外，这些概率计算好后，还得保存下来，因此，存储这些信息也需要很大的内存开销.

此外，**这些参数如何计算呢?** 常见的方法有 n-gram 模型、决策树、最大熵模型、最大熵马尔科夫模型、条件随机场、神经网络等方法. 本文只讨论 **n-gram 模型**和**神经网络**两种方法. 首先来看看 n-gram 模型.

关闭

§3.2 n-gram 模型

考虑 $p(w_k|w_1^{k-1})$ $(k>1)$ 的近似计算. 利用 Bayes 公式, 有

$$p(w_k|w_1^{k-1}) = \frac{p(w_1^k)}{p(w_1^{k-1})},$$

根据大数定理, 当语料库足够大时, $p(w_k|w_1^{k-1})$ 可近似地表示为

$$p(w_k|w_1^{k-1}) \approx \frac{count(w_1^k)}{count(w_1^{k-1})}, \tag{3.2}$$

其中 $count(w_1^k)$ 和 $count(w_1^{k-1})$ 分别表示词串 w_1^k 和 w_1^{k-1} 在语料中出现的次数. 可想而知, 当 k 很大时, $count(w_1^k)$ 和 $count(w_1^{k-1})$ 的统计将会多么耗时.

从公式 (3.1) 可以看出: 一个词出现的概率与它前面的所有词都相关. 如果假定一个词出现的概率只与它前面固定数目的词相关呢? 这就是 **n-gram 模型**的基本思想, 它作了一个 $n-1$ 阶的 **Markov 假设**, 认为一个词出现的概率就只与它前面的 $n-1$ 个词相关, 即

$$p(w_k|w_1^{k-1}) \approx p(w_k|w_{k-n+1}^{k-1}),$$

于是, (3.2) 就变成了

$$p(w_k|w_1^{k-1}) \approx \frac{count(w_{k-n+1}^k)}{count(w_{k-n+1}^{k-1})}. \tag{3.3}$$

以 $n=2$ 为例, 就有

$$p(w_k|w_1^{k-1}) \approx \frac{count(w_{k-1}, w_k)}{count(w_{k-1})}.$$

这样一简化, 不仅使得单个参数的统计变得更容易 (统计时需要匹配的词串更短), 也使得参数的总数变少了.

那么, n-gram 中的参数 n 取多大比较合适呢? 一般来说, n 的选取需要同时考虑计算复杂度和模型效果两个因素.

表 1 模型参数数量与 n 的关系

n	模型参数数量
1 (unigram)	2×10^5
2 (bigram)	4×10^{10}
3 (trigram)	8×10^{15}
4 (4-gram)	16×10^{20}

在**计算复杂度**方面, 表 1 给出了 n-gram 模型中模型参数数量随着 n 的逐渐增大而变化的情况, 其中假定词典大小 $N=200000$ (汉语的词汇量大致是这个量级). 事实上, 模型参数的量级是 N 的指数函数 $(O(N^n))$, 显然 n 不能取得太大, 实际应用中最多的是采用 $n=3$ 的三元模型.

在**模型效果**方面, 理论上是 n 越大, 效果越好. 现如今, 互联网的海量数据以及机器性能的提升使得计算更高阶的语言模型 (如 $n>10$) 成为可能, 但需要注意的是, 当 n 大到一定程度时, 模型效果的提升幅度会变小. 例如, 当 n 从 1 到 2, 再从 2 到 3 时, 模型的效果上升显著, 而从 3 到 4 时, 效果的提升就不显著了 (具体可参考吴军在《数学之美》中的相关章节). 事实上, 这里还涉及到一个**可靠性**和**可区别性**的问题, 参数越多, 可区别性越好, 但同时单个参数的实例变少从而降低了可靠性, 因此需要在可靠性和可区别性之间进行折中.

另外, n-gram 模型中还有一个叫做**平滑化**的重要环节. 回到公式 (3.3), 考虑两个问题:

1. 若 $count(w_{k-n+1}^k)=0$, 能否认为 $p(w_k|w_1^{k-1})$ 就等于 0 呢?

2. 若 $count(w_{k-n+1}^{k}) = count(w_{k-n+1}^{k-1})$, 能否认为 $p(w_k|w_1^{k-1})$ 就等于 1 呢?

显然不能! 但这是一个无法回避的问题, 哪怕你的语料库有多么大. 平滑化技术就是用来处理这个问题的, 这里不展开讨论, 具体可参考 [11].

总结起来, n-gram 模型是这样一种模型, 其主要工作是在语料中统计各种词串出现的次数以及平滑化处理. 概率值计算好之后就存储起来, 下次需要计算一个句子的概率时, 只需找到相关的概率参数, 将它们连乘起来就好了.

然而, 在机器学习领域有一种通用的招数是这样的: 对所考虑的问题建模后先为其构造一个目标函数, 然后对这个目标函数进行优化, 从而求得一组最优的参数, 最后利用这组最优参数对应的模型来进行预测.

对于统计语言模型而言, 利用**最大似然**, 可把目标函数设为

$$\prod_{w\in\mathcal{C}} p(w|Context(w)).$$

其中 $\mathcal{C}$ 表示语料 (Corpus), $Context(w)$ 表示词 w 的**上下文** (Context), 即 w 周边的词的集合. 当 $Context(w)$ 为空时, 就取 $p(w|Context(w)) = p(w)$. 特别地, 对于前面介绍的 n-gram 模型, 就有 $Context(w_i) = w_{i-n+1}^{i-1}$.

注 3.1 *语料 $\mathcal{C}$ 和词典 $\mathcal{D}$ 的区别: 词典 $\mathcal{D}$ 是从语料 $\mathcal{C}$ 中抽取出来的, 不存在重复的词; 而语料 $\mathcal{C}$ 是指所有的文本内容, 包括重复的词.*

当然, 实际应用中常采用**最大对数似然**, 即把目标函数设为

$$\mathcal{L} = \sum_{w\in\mathcal{C}} \log p(w|Context(w)), \tag{3.4}$$

然后对这个函数进行最大化.

从 (3.4) 可见, 概率 $p(w|Context(w))$ 已被视为关于 w 和 $Context(w)$ 的**函数**, 即

$$p(w|Context(w)) = F(w, Context(w), \theta),$$

其中 θ 为**待定参数集**. 这样一来, 一旦对 (3.4) 进行优化得到最优参数集 θ^* 后, F 也就唯一被确定了, 以后任何概率 $p(w|Context(w))$ 就可以通过函数 $F(w, Context(w), \theta^*)$ 来计算了. 与 n-gram 相比, 这种方法不需要 (事先计算并) 保存所有的概率值, 而是通过直接计算来获取, 且通过选取合适的模型可使得 θ 中参数的个数远小于 n-gram 中模型参数的个数.

很显然, 对于这样一种方法, 最关键的地方就在于**函数 F 的构造**了. 下一小节将介绍一种通过神经网络来构造 F 的方法. 之所以特意介绍这个方法, 是因为它可以视为 word2vec 中算法框架的前身或者说基础.

§3.3 神经概率语言模型

本小节介绍 **Bengio** 等人在文《A neural probabilistic language model. Journal of Machine Learning Research》(2003) 中提出的一种神经概率语言模型 ([2]). 该模型中用到了一个重要的工具 — **词向量**.

什么是词向量呢? 简单来说就是, 对词典 $\mathcal{D}$ 中的任意词 w, 指定一个固定长度的实值向量 $\mathbf{v}(w) \in \mathbb{R}^m$, $\mathbf{v}(w)$ 就称为 w 的词向量, m 为词向量的长度. 关于词向量的进一步理解将放到下一小节来讲解.

既然是神经概率语言模型, 其中当然要用到一个神经网络啦. 图 4 给出了这个神经网络的结构示意图, 它包括四个层: **输入** (Input) 层、**投影** (Projection) 层、**隐藏** (Hidden) 层和**输出** (Output) 层. 其中 W, U 分别为投影层与隐藏层以及隐藏层和输出层之间的权值矩阵, $\mathbf{p}$, $\mathbf{q}$ 分别为隐藏层和输出层上的偏置向量.

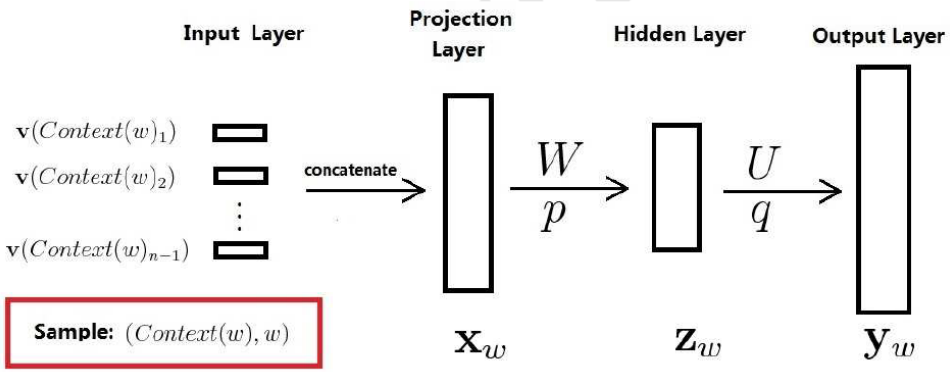

图 4 神经网络结构示意图

注 3.2 当提及文 [2] 中的神经网络时, 人们更多将其视为如图 5 所示的三层结构. 本文将其描述为如图 4 所示的四层结构, 一方面是便于描述, 另一方面是便于和 word2vec 中使用的网络结构进行对比.

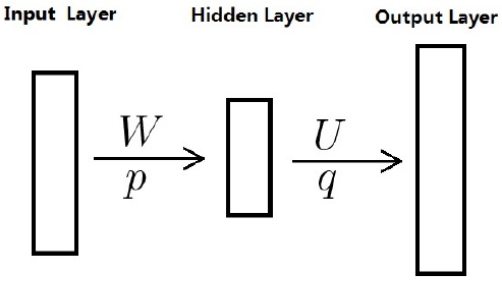

图 5 三层神经网络结构示意图

注 3.3 作者在文 [2] 中还考虑了投影层和输出层的神经元之间有边相连的情形，因而也会多出一个相应的权值矩阵，本文忽略了这种情形，但这并不影响对算法本质的理解. 在数值实验中，作者发现引入投影层和输出层之间的权值矩阵虽然不能提高模型效果，但可以减少训练的迭代次数.

对于语料 $\mathcal{C}$ 中的任意一个词 w, 将 $Context(w)$ 取为其前面的 $n-1$ 个词 (类似于 n-gram), 这样二元对 $(Context(w), w)$ 就是一个**训练样本**了. 接下来, 讨论样本 $(Context(w), w)$ 经过如图 4 所示的神经网络时是如何参与运算的. 注意, 一旦语料 $\mathcal{C}$ 和词向量长度 m 给定后, 投影层和输出层的规模就确定了, 前者为 $(n-1)m$, 后者为 $N = |\mathcal{D}|$ 即语料 $\mathcal{C}$ 的词汇量大小. 而隐藏层的规模 n_h 是可调参数由用户指定.

为什么投影层的规模是 $(n-1)m$ 呢? 因为输入层包含 $Context(w)$ 中 $n-1$ 个词的词向量, 而投影层的向量 $\mathbf{x}_w$ 是这样构造的: 将输入层的 $n-1$ 个词向量按顺序首尾相接地拼起来形成一个长向量, 其长度当然就是 $(n-1)m$ 了. 有了向量 $\mathbf{x}_w$, 接下来的计算过程就很平凡了, 具体为

$$\begin{cases} \mathbf{z}_w = \tanh(W\mathbf{x}_w + \mathbf{p}), \\ \mathbf{y}_w = U\mathbf{z}_w + \mathbf{q}, \end{cases} \tag{3.5}$$

其中 tanh 为**双曲正切函数**, 用来做隐藏层的**激活函数**, 上式中, tanh 作用在向量上表示它作用在向量的每一个分量上.

注 3.4 有读者可能要问: 对于语料中的一个给定句子的前几个词, 其前面的词不足 $n-1$ 个怎么办? 此时, 可以人为地添加一个 (或几个) 填充向量就可以了, 它们也参与训练过程.

经过上述两步计算得到的 $\mathbf{y}_w = (y_{w,1}, y_{w,2}, \cdots, y_{w,N})^\top$ 只是一个长度为 N 的向量, 其分量不能表示概率. 如果想要 $\mathbf{y}_w$ 的分量 $y_{w,i}$ 表示当上下文为 $Context(w)$ 时下一个词恰为词典 $\mathcal{D}$ 中第 i 个词的概率, 则还需要做一个 **softmax** 归一化, 归一化后, $p(w|Context(w))$ 就可以表示为

$$p(w|Context(w)) = \frac{e^{y_{w,i_w}}}{\sum_{i=1}^{N} e^{y_{w,i}}}, \tag{3.6}$$

其中 i_w 表示词 w 在词典 $\mathcal{D}$ 中的索引.

公式 (3.6) 给出了概率 $p(w|Context(w))$ 的函数表示, 即找到了上一小节中提到的函数 $F(w, Context(w), \theta)$, 那么其中待确定的参数 θ 有哪些呢? 总结起来, 包括两部分

- 词向量: $\mathbf{v}(w) \in \mathbb{R}^m, w \in \mathcal{D}$ 以及填充向量.
- 神经网络参数: $W \in \mathbb{R}^{n_h \times (n-1)m}$, $\mathbf{p} \in \mathbb{R}^{n_h}$; $U \in \mathbb{R}^{N \times n_h}$, $\mathbf{q} \in \mathbb{R}^N$,

这些参数均通过训练算法得到. 值得一提的是, 通常的机器学习算法中, 输入都是已知的, 而在上述神经概率语言模型中, 输入 $\mathbf{v}(w)$ 也需要通过训练才能得到.

接下来, 简要地分析一下上述模型的运算量. 在如图 4 所示的神经网络中, 投影层、隐藏层和输出层的规模分别为 $(n-1)m$, n_h, N, 依次看看其中涉及的参数:

(1) n 是一个词的上下文中包含的词数, 通常不超过 5;
(2) m 是词向量长度, 通常是 $10^1 \sim 10^2$ 量级;
(3) n_h 由用户指定, 通常不需取得太大, 如 10^2 量级;
(4) N 是语料词汇量的大小, 与语料相关, 但通常是 $10^4 \sim 10^5$ 量级.

再结合 (3.5) 和 (3.6), 不难发现, 整个模型的大部分计算集中在隐藏层和输出层之间的矩阵向量运算, 以及输出层上的 softmax 归一化运算. 因此后续的相关研究工作中, 有很多是针对这一部分进行优化的, 其中就包括了 word2vec 的工作.

与 n-gram 模型相比, 神经概率语言模型有什么**优势**呢? 主要有以下两点:

1. 词语之间的相似性可以通过词向量来体现.

   举例来说, 如果某个 (英语) 语料中 $S_1 =$ "A dog is running in the room" 出现了 10000 次, 而 $S_2 =$ "A cat is running in the room" 只出现了 1 次. 按照 n-gram 模型的做法, $p(S_1)$ 肯定会远大于 $p(S_2)$. 注意, S_1 和 S_2 的唯一区别在于 dog 和 cat, 而这两个词无论是句法还是语义上都扮演了相同的角色, 因此, $p(S_1)$ 和 $p(S_2)$ 应该很相近才对.

   然而, 由神经概率语言模型算得的 $p(S_1)$ 和 $p(S_2)$ 是大致相等的. 原因在于: (1) 在神经概率语言模型中假定了"相似的"的词对应的词向量也是相似的; (2) 概率函数关于词向量是光滑的, 即词向量中的一个小变化对概率的影响也只是一个小变化. 这样一来, 对于下面这些句子

   ```
   A dog is running in the room
   A cat is running in the room
   The cat is running in a room
   A dog is walking in a bedroom
   The dog was walking in the room
      ...
   ```

   只要在语料库中出现一个, 其他句子的概率也会相应地增大.
2. 基于词向量的模型自带平滑化功能 (由 (3.6) 可知, $p(w|Context(w)) \in (0,1)$ 不会为零), 不再需要像 n-gram 那样进行额外处理了.

最后, 我们回过头来想想, 词向量在整个神经概率语言模型中扮演了什么角色呢? 训练时, 它是用来帮助构造目标函数的辅助参数, 训练完成后, 它也好像只是语言模型的一个副产品. 但这个副产品可不能小觑, 下一小节将对其作进一步阐述.

§3.4 词向量的理解

通过上一小节的讨论, 相信大家对词向量已经有一个初步的认识了. 接下来, 对词向量做进一步介绍.

在 NLP 任务中, 我们将自然语言交给机器学习算法来处理, 但机器无法直接理解人类的语言, 因此首先要做的事情就是将语言数学化, 如何对自然语言进行数学化呢? 词向量提供了一种很好的方式.

一种最简单的词向量是 **one-hot representation**, 就是用一个很长的向量来表示一个词, 向量的长度为词典 $\mathcal{D}$ 的大小 N, 向量的分量只有一个 1, 其它全为 0, 1 的位置对应该词在词典中的索引. 但这种词向量表示有一些缺点, 如容易受维数灾难的困扰, 尤其是将其用于 Deep Learning 场景时; 又如, 它不能很好地刻画词与词之间的相似性.

另一种词向量是 **Distributed Representation**, 它最早是 Hinton 于 1986 年提出的 ([1]), 可以克服 one-hot representation 的上述缺点. 其基本想法是: 通过训练将某种语言中的每一个词映射成一个固定长度的短向量 (当然这里的 "短" 是相对于 one-hot representation 的 "长" 而言的), 所有这些向量构成一个词向量空间, 而每一向量则可视为该空间中的一个点, 在这个空间上引入 "距离", 就可以根据词之间的距离来判断它们之间的 (词法、语义上的) 相似性了. word2vec 中采用的就是这种 Distributed Representation 的词向量.

为什么叫做 Distributed Representation? 很多人问到这个问题. 我的一个理解是这样的: 对于 one-hot representation, 向量中只有一个非零分量, 非常集中 (有点孤注一掷的感觉); 而对于 Distributed Representation, 向量中有大量非零分量, 相对分散 (有点风险平摊的感觉), 把词的信息分布到各个分量中去了. 这一点, 跟并行计算里的分布式并行很像.

为更好地理解上述思想, 我们来举一个通俗的例子.

例 3.2 假设在二维平面上分布有 a 个不同的点, 给定其中的某个点, 现在想在平面上找到与这个点最相近的一个点.

我们是怎么做的呢? 首先, 建立一个直角坐标系, 基于该坐标系, 其上的每个点就唯一地对应一个坐标 (x, y); 接着引入欧氏距离; 最后分别计算这个点与其他 $a-1$ 个点之间的距离, 对应最小距离值的那个 (或那些) 点便是我们要找的点了.

上面的例子中, 坐标 (x, y) 的地位就相当于词向量, 它用来将平面上一个点的位置在数学上作量化. 坐标系建立好以后, 要得到某个点的坐标是很容易的. 然而, 在 NLP 任务中, 要得到词向量就复杂得多了, 而且词向量并不唯一, 其质量依赖于训练语料、训练算法等因素.

如何获取词向量呢? 有很多不同模型可用来估计词向量, 包括有名的 LSA (Latent Semantic Analysis) 和 LDA (Latent Dirichlet Allocation). 此外, 利用神经网络算法也是一种常用的方法, 上一小节介绍的神经概率语言模型就是一个很好的实例. 当然, 在那个模型中, 目标是生成语言模型, 词向量只是一个副产品. 事实上, 大部分情况下, 词向量和语言模型都是捆绑在一起的, 训练完成后两者同时得到. 用神经网络来训练语言模型的思想最早由百度 IDL (深度学习研究院) 的**徐伟**提出 ([14]). 这方面最经典的文章要数 **Bengio** 于 2003 年发表在 JMLR 上的《A Neural Probabilistic Language Model》, 其后有一系列相关的研究工作, 其中也包括谷歌 **Tomas Mikolov** 团队的 word2vec.

Task	Benchmark		Performance	Timing (s)
Part of Speech (POS)	(Toutanova et al, 2003)	(Accuracy)	97.29%	3
Chunking (CHK)	CoNLL 2000	(F1)	94.32%	2
Name Entity Recognition (NER)	CoNLL 2003	(F1)	89.59%	2
Semantic Role Labeling (SRL)	CoNLL 2005	(F1)	75.49%	36
Syntactic Parsing (PSG)	Penn Treebank	(F1)	87.92%	74

图 6 SENNA performance in per-word accuracy for POS, and F1 score for all the other tasks. Timing corresponds to the time needed by SENNA to pass over the given test data set (Macbook Pro i7, 2.8GHz, Intel MKL). For PSG, F1 score is the one over all sentences.

一份好的词向量是很有价值的, 例如, **Ronan Collobert** 团队在软件包 **SENNA** ([12]) 中利用词向量进行了 POS、CHK、NER 等任务, 且取得了不错的效果 (见图 6 中的表格). 最近了解到词向量在机器翻译领域的一个应用 ([13]), 报道是这样的:

> 谷歌的 Tomas Mikolov 团队开发了一种词典和术语表的自动生成技术, 能够把一种语言转变成另一种语言. 该技术利用数据挖掘来构建两种语言的结构模型, 然后加以对比. 每种语言词语之间的关系集合即“语言空间”, 可以被表征为数学意义上的向量集合. 在向量空间内, 不同的语言享有许多共性, 只要实现一个向量空间向另一个向量空间的映射和转换, 语言翻译即可实现. 该技术效果非常不错, 对英语和西语间的翻译准确率高达 90%.

文 [5] 在引言中介绍算法原理时举了一个简单的例子, 可以帮助我们更好地理解词向量的工作原理, 特将其介绍如下.

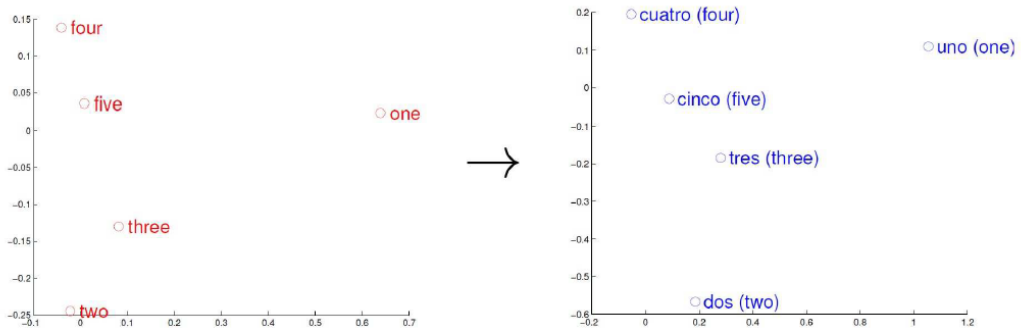

图 7 五个词在两个向量空间中的位置 (左: E 右: S)

考虑英语和西班牙语两种语言, 通过训练分别得到它们对应的词向量空间 E(nglish) 和 S(panish). 从英语中取出五个词 one, two, three, four, five, 设其在 E 中对应的词向量分别为 u_1, u_2, u_3, u_4, u_5, 为方便作图, 利用主成分分析 (PCA) 降维, 得到相应的二维向量 v_1, v_2, v_3, v_4, v_5, 在二维平面上将这五个点描出来, 如图 7 左图所示.

类似地, 在西班牙语中取出 (与 one, two, three, four, five 对应的) uno, dos, tres, cuatro, cinco, 设其在 S 中对应的词向量分别为 s_1, s_2, s_3, s_4, s_5, 用 PCA 降维后的二维向量分别为 t_1, t_2, t_3, t_4, t_5, 将它们在二维平面上描出来 (可能还需作适当的旋转), 如图 7 右图所示.

观察左、右两幅图, 容易发现: 五个词在两个向量空间中的相对位置差不多, 这说明两种不同语言对应向量空间的结构之间具有相似性, 从而进一步说明了在词向量空间中利用距离刻画词之间相似性的合理性.

注意, 词向量只是针对“词”来提的, 事实上, 我们也可以针对更细粒度或更粗粒度来进行推广, 如**字向量** ([7]), **句子向量**和**文档向量** ([6]), 它们能为字、句子、文档等单元提供更好的表示.

参考文献

[1] David E Rumelhart, Geoffrey E Hintont, and Ronald J Williams. **Learning representations by backpropagating errors.** Nature, 323(6088):533-536, 1986.

[2] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. **A neural probabilistic language model.** Journal of Machine Learning Research (JMLR), 3:1137-1155, 2003.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. **Efficient Estimation of Word Representations in Vector Space.** arXiv:1301.3781, 2013.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. **Distributed Representations of Words and Phrases and their Compositionality.** arXiv:1310.4546, 2013.

[5] Tomas Mikolov, Quoc V. Le, Ilya Sutskever. **Exploiting Similarities among Languages for Machine Translation.** arXiv:1309.4168v1, 2013.

[6] Quoc V. Le, Tomas Mikolov. **Distributed Representations of Sentences and Documents.** arXiv:1405.4053, 2014.

[7] Xiaoqing Zheng, Hanyang Chen, Tianyu Xu. **Deep Learning for Chinese Word Segmentation and POS tagging.** Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 647-657.

[8] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu and Pavel Kuksa. **Natural Language Processing (Almost) from Scratch.** Journal of Machine Learning Research (JMLR), 12:2493-2537, 2011.

[9] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. The Journal ofMachine Learning Research, 13:307-361, 2012.

[10] 百度百科中的“哈夫曼树”词条.

[11] 吴军.**《数学之美》**. 人民邮电出版社, 2012.

[12] http://ml.nec-labs.com/senna/

[13] http://www.looooker.com/archives/5621

[14] licstar. **Deep Learning in NLP (一) 词向量和语言模型**. http://licstar.net/archives/328

[15] **深度学习 word2vec 笔记之基础篇**. http://blog.csdn.net/mytestmy/article/details/26961315

上一篇 word2vec 中的数学原理详解（二）预备知识

下一篇 word2vec 中的数学原理详解（四）基于 Hierarchical Softmax 的模型

相关文章推荐

- word2vec 中的数学原理详解（三）背景知识
- 【直播】70天软考冲刺计划--任铄
- word2vec原理解析
- 【直播】打通Linux脉络 进程、线程、调度--宋宝华
- word2vec中的数学原理详解
- 【直播】机器学习之凸优化--马博士
- word2vec数学原理
- 【套餐】MATLAB基础+MATLAB数据分析与统计-…
- word2vec 中的数学原理详解（五）基于 Negative…
- 【课程】3小时掌握Docker最佳实战--徐西宁
- Word2Vec数据集
- 【课程】深度学习基础与TensorFlow实践--AI100
- word2vec 中的数学原理详解（二）预备知识
- word2vec工具下载
- word2vec-master
- word2vec源文件

查看评论

34楼 PJ-Javis 5天前 15:34发表

> 引用"ayhx94"的评论：
> 公式里有一个大坑啊，，w1既表示句子里的一个单词，加上右上角的标号后又表示T长度的一个句子，结果在下...

这的确是个坑，我也掉进去了

33楼 ayhx94 2017-08-10 15:45发表

公式里有一个大坑啊，，w1既表示句子里的一个单词，加上右上角的标号后又表示T长度的一个句子，结果在下面贝叶斯公式里，一开始我总以为那是右上角的标号是次方，还说怎么能推出这个，，

32楼 rkx111111 2017-08-06 18:15发表

博主可以发我一份pdf文档吗？邮箱：2622749201@qq.com，谢谢博主啦！

31楼 joycema200 2017-07-18 13:39发表

楼主写得非常好，可以发一份pdf吗~~ andersonma@126.com 谢谢啦！

30楼 IT路上的苦行僧 2017-07-17 21:00发表

写得深入浅出，求楼主发一份pdf拜读，hysfwjr@163.com

29楼 shaozhe2016 2017-07-15 11:38发表

您好，您在3.4节说词向量是上述的神经概率模型的副产品，但是上述的神经概率模型的输入层，已经是n-1个词的词向量了，并不是由该神经概率模型产生的啊

Re: shaozhe2016 2017-07-15 11:40发表

回复shaozhe2016：哦，我注意到了，输入层的词向量也要通过网络训练得到的，不好意思

28楼 RookieJackokie 2017-07-14 08:33发表

求楼主发一份pdf拜读，谢谢啦 jackokie@qq.ocm

27楼 m0_37052320 2017-06-14 16:36发表

词向量和one-hot完全没有关系么？对输入的词向量的初始化同样存在疑问，直接随机初始化？反正都是要训练的

26楼 m0_37052320 2017-06-14 16:35发表

回复qifeiyang112358：词向量和one-hot完全没有关系么？对输入的词向量的初始化同样存在疑问，直接随机初始化？反正都是要训练的

25楼 又有口红 2017-06-06 14:47发表

求完整的PDF文档，1139231246@qq.com

24楼 qq_24953899 2017-05-02 21:17发表

同求pdf文档，邮箱1005703344@qq.com

关闭

23楼 qq_24953899 2017-05-02 16:01发表

同求pdf文档，邮箱1005703344@qq.com

22楼 qq_18857415 2016-12-23 11:57发表

目标函数中，x属于集合C。既然C是语料库，是有重复的，就违背了集合的定义的互异性。是否应该改为x属于字典D？

21楼 Alistarhu 2016-10-24 17:04发表

楼主总结的很好！能发我一份PDF，十分感谢！hulei557@163.com

20楼 _蟾宫客- 2016-09-28 20:38发表

博主能发一个pdf吗？谢谢！
1055970018@qq.com

19楼 __xwzhong___ 2016-07-23 16:54发表

博主，求一份pdf，thanks
727134010@qq.com

18楼 zgkdzw 2016-02-03 22:43发表

博主，你的文章写得很好，可以发一份PDF version给我吗？想打印下看看，多谢！zgkdzw@gmail.com

17楼 shixiang08abc 2016-01-04 19:45发表

楼主！同求一份PDF，shixiangabc@163.com，非常感谢！！

16楼 shixiang08abc 2016-01-04 19:25发表

可以给我发一份PDF文档吗？shixiangabc@163.com，谢谢啦！

15楼 Cathy1272014 2015-10-17 17:00发表

博主，可否发一下 P D F 文档，邮箱：570571711@qq.com

14楼 ECNU_zwq 2015-10-14 21:56发表

博主，Bengio的文章中的样本输入，是语料中遍历的所有的(Context(w), w)中的Context(W)的向量相互连接成的向量
输出是一个词典长度V，大小的向量，在softmax归一化之后，表征的是P(w | context(w))，也就是词典中每个词在Context(w)后出现的概率
这里的Context(w)是输入中的Context对吗？
对每个样本都会进过神经网络算出一组概率
我们取出这组概率中的对应了样本中w的那个P(w | context(w))
然后对每个样本都算出这个唯一的值
然后将所有这些值相乘才是目标函数的值吧？
我不知道我理解的对不对。。。

13楼 ECNU_zwq 2015-10-14 21:55发表

博主，Bengio的文章中的样本输入，是语料中遍历的所有的(Context(w), w)中的Context(W)的向量相互连接成的向量
输出是一个词典长度V，大小的向量，在softmax归一化之后，表征的是P(w | context(w))，也就是词典中每个词在Context(w)后出现的概率
这里的Context(w)是输入中的Context对吗？
对每个样本都会进过神经网络算出一组概率
我们取出这组概率中的对应了样本中w的那个P(w | context(w))
然后对每个样本都算出这个唯一的值
然后将所有这些值相乘才是目标函数的值吧？
我不知道我理解的对不对。。。

Re: qifeiyang112358 2016-06-03 14:58发表

回复ECNU_zwq：有一点疑问，是输入的词向量是怎么初始化的，对应（context（w),w)的对应n-gram的每个单词的最早的词向量，相比较one-hot 表示，这里m的维度，以及每个维度的大小怎么得到的（在输入神经网络进行计算拟合之前）？

Re: m0_37052320 2017-06-14 16:34发表

回复qifeiyang112358：词向量和one-hot完全没有关系么？对输入的词向量的初始化同样存在疑问，直接随机初始化？反正都是要训练的

12楼 piao00lingping 2015-06-29 11:11发表

博主您好，我想请教一下：LSA、LDA如何训练得到词向量呢？有没有相关方面的详细介绍，可以发我一份吗？我的邮箱是muziqingqing@yahoo.com,谢谢！

11楼 victormmd 2015-04-19 20:54发表

关闭

博主，博文写的很精彩，言简意赅，我想求一份pdf文档只做学术研究，不做其他用途，真心谢谢啦
我的邮箱：351868656@qq.com

10楼 novice_xqjin 2015-04-04 21:15发表

博主，您好，我不太理解您在博文中提到的： 神经网络语言模型假设相似的词，对应的词向量相同 ， 模型如何能够做到？ 或者您在哪篇论文中看到有相关的论述呢？ 谢谢您；

9楼 extends_die 2015-02-17 13:03发表

你好，n - gram模型中的可靠性和可区别性具体是指什么，没有看明白，能帮我讲讲吗？谢谢了～

8楼 taptree 2015-01-04 00:14发表

博主您好，能发一份pdf文档到我邮箱吗？
taptree@163.com，谢谢！祝新年快乐！

7楼 moran123321 2014-09-18 10:40发表

博主您好，如果方便，请您也给我发一份这个文档。邮箱是louqi_hdu#163.com(#用@替代)

6楼 rockeinstein2 2014-09-04 11:30发表

楼主！同求一份PDF，rock.cloud0521@gmail.com，非常感谢！！

5楼 jintaiwei 2014-09-02 10:33发表

求博主也发我一份pdf文档！742195993@qq.com 万分感谢~

4楼 jintaiwei 2014-09-01 17:26发表

博主 问一个问题：LSA LDA得到词向量这个怎么理解？ LDA最后得到的是表示文档的一个向量，以及一堆主题，有点不明词向量这点如何体现？

Re: qifeiyang112358 2016-06-03 14:53发表

回复jintaiwei：新入坑新人，求资料转发一下：1007451765@qq.com。先谢前辈了。

Re: 皮果提 2014-09-02 18:35发表

回复jintaiwei：已回邮件，请查收！

Re: jintaiwei 2014-09-02 18:37发表

回复peghoty：已查收。我昨天去查了一下，LSA LDA的词向量在粒度上不一样，应该就是指的文档的主题分布。。谢谢

3楼 小飞_侠 2014-08-26 23:52发表

博主，能发份pdf的吗？邮箱是chongqing spf@gmail. com，谢谢

Re: 小飞_侠 2014-08-29 17:45发表

回复a6210575：不好意思，邮箱写错啦，多了一个空格，chongqingspf@gmail.com，谢谢

2楼 nonorth 2014-08-25 11:21发表

可以给我发一份PDF文档吗？nonorth@sina.com，谢谢啦！

1楼 youliah 2014-08-21 23:45发表

可以给我发一份PDF文档吗？ymlinbh@163.com，谢谢啦！

关闭