博客 | Word2Vec 学习心得

机器学习研究组订阅号 2018-08-17

好嘛博主食言了。不过本文没什么干货，主要是前后看了大概一个星期，反复去读源码和解读文章，终于感觉这东西不那么云山雾罩了。同时也发现网上很多材料有点扯淡，99% 的博文不过是把别人的东西用自己的话说一下，人云亦云。好多人自己理解错了而不自知，实在是误人误己。

我也不敢说理解得有多深，下面的内容甚至可能有自相矛盾的地方，所以阅读本文时请一定擦亮眼睛，认真思考。

源码才是根本，作者那两篇论文感觉参考价值也不高。说到底，Machine Learning/Deep Learning 的价值在于实践，而实际开发的应用中经过大量的 tricks 之后，代码跟论文推导、实验可能相去甚远。

Data Mining 是一门实验科学，编程实现、实验所用的数据集都可能对假设和结论产生无法预知的影响，希望各位时刻牢记。

0 一段前言

个人觉得学 Word2Vec 有几种路径。如果你代码能力足够，建议直接读作者的原始论文和 Rong Xin 的解释篇，然后找一些比较高质量的源码解析作为辅助，这样效率比较高。如果不行，那就得多读一些东西，比如来斯为的博文和博士论文、网上中文的英文的详解博客，建立一组整体概念，然后对照源码来看。

前者的典型见：

http://www.cnblogs.com/neopenx/p/4571996.html

后者的典型是皮果提的“Word2Vec 数学原理”系列。正如前者所说，数学熟练如皮果提也在理解 Skip-Gram 时犯了先入为主的错误；而强如 Physcal 也会一时疏忽把 Huffman 树记成了满二叉树。

所以再三强调，要批判地看本文和网上一切的解读文章。感慨一下，我因为这个东西看了不下三十篇中文的英文的博文，皮果提的“数学原理”颇有夸大其词之嫌，但已经算是相当有深度的，像其他一些所谓解读不过是把论文的图复制过来然后用自己的理解随便说两句，所以你能看到大量相关博文介绍 CBOW、Skip-Gram 俩模型但往往浅尝辄止。

理解到 Physcal 博文即 http://www.cnblogs.com/neopenx/p/4571996.html 这篇程度的人恐怕不足千分之一，包括但不限于皮果提、网易的w2v笔记和国外一些看起来很好看的博文。大多数人只是简单看了一下论文，好一点儿的如来斯为的博文和博士论文，也只是做了发展梳理和一些后续实验，感觉并没有结合代码本身对论文的提法做一些重构。

那篇“秒懂词向量”的文章也不过是刚入门的程度，感觉比我强不到哪儿去……

对了，苏剑林的科学空间建议看一下，我觉得这位在 NLP 方面的理解也很厉害，可能跟数学出身有关吧。

本文将根据自己的理解，把 Word2Vec 这款开源工具做一点梳理，可能更适合厘清一些误解而不适合新手入门。有需要可以参考上面的说明自己找资料学习，文中不会过多解释。

1 几个概念

1.1 Word2Vec

Word2Vec 是 Google 开源的一款词向量训练工具，特点是效率高，据称可"单机在一天内训练完一个包含 16 亿单词的训练集"。它是对以往的 NNLM 的改进，相关内容可参考来斯为的博客和博士论文，介绍比较详细。

但 Word2Vec 并不是最佳的词向量训练工具，它流行的原因是 Google 光环加成和相对高效。更新的 FastText 速度更快，而且论文也有去了 FB 的 Mikolov 挂名。

实际上就"词的分布式表示（DR）"问题有三类模型，据来斯为的博士论文，包括：

基于矩阵的DR，包括 tf-idf based LSA，GloVe 等

基于聚类的DR，如布朗聚类

基于NN的DR即 Word Vector，可以对上下文、上下文与目标词关系进行建模

"分布表示（distributional representation）"意为"通过上下文表示语义"。根据实验，三类模型之间不存在谁好谁坏，参数不同、语料不同时表现（"精确程度"的各类衡量指标和训练速度）相差很大。而且三类模型之间存在一定的联系，如来斯为证明了 Skip-gram + Negtive Sampling 与 Glove 的等价性。可以说不同的算法更多是对不同的指标的权衡，应当根据实际情况选择合适的方案。希望读者不要忘记这一点。

1.2 CBOW & Skip-gram

完整的 NNLM 最早由 Bengio（2003） 提出，可以看作四层网络。由于隐藏层有非线性、输出层是 Softmax （输出到词表大小），计算量非常大。

Mikolov 提出的 CBOW 、 Skip-Gram 两个模型，是对过去 NNLM 的大幅简化，除输出仍为 Softmax 外，只有一层线性连接。Hierarchical Softmax 和 Negative Sampling 则是进一步加速训练的方法。总之 Mikolov 取胜的最主要因素就是训练速度，在给出过得去的准确度的情况下使用很多激进方法加速。这方面网上也有很多对比，不再赘述。

另外参见 Physcal 的文章，Skip-Gram 不是 CBOW 的对称模型，而是一个精度更高的版本。"CBOW 是上下文预测当前词、Skip-Gram 是当前词预测上下文"的说法并不准确，只是一种暧昧的形容而非实

事。作为专业人士要尽量避免“用简单形容来概括复杂操作/概念”的尝试，用来向外行人解释或者加强自己的记忆尚可，当了真就不对。想知道真实情况，应该去读源码。

来斯为的博士论文里提了一句大意是“实际上两个模型都是根据上下文预测当前词”，只不过他的解释是“两种模型都要遍历全文”，我也没看懂到底哪个解释更高明些……

word2vec.c 中，CBOW 是 422-482 行，SG 是 482-530 行。

可以看到，CBOW 是先在一个窗口内把 Context(w) 算完（求和-平均）做 FF 、BP，更新窗口词的参数，而 Skip-Gram 在窗口内则是逐个更新【目标词和窗口中某个词】组成的词对，每个词对更新一次参数。两段代码对比着看会更明显。

也就是说， CBOW 的 FF 和 BP 都是一把梭，窗口里的词算完 FF 再算 BP；而 Skip-Gram 更小心，一个【词对】算一次 FF 和 BP。这就是 SG 比 CBOW 要慢得多得多的原因。效果如何呢？网上一般的黑盒经验就是所谓大语料用 CBOW，小语料用 SG 或者 CBOW 对高频词友好、 SG 对低频词友好了。说白了还是数据说话，试了才知道。

我觉得更合适的类比是，和两个模型代表两种训练方法：CBOW 像 mini-batch，SG 像 online。这样精度更高的含义好像就不难理解了吧?

Negative Sampling、Hierarchical Softmax 两种模式的计算量看起来差不多，但由于 HS 采用了 Huffman 编码，高频词路径更短，所以对一般语料（大量高频词）的训练速度会更快一些。

再用 Physcal 的话解释一遍：CBOW 直接把上下文平均输入进去更新目标词，相当于把几个词看成了一个词；而 Skip-Gram 仍然以词对的形式将窗口内的词逐个更新。

所以，虽然 Mikolov 论文里画的图是对称的，两个模型实际并非镜像。Skip-gram 的图画成 1 对 n 是结果的一种呈现而不是计算过程的说明。

1.3 词向量 & 词嵌入

本条参考 Wikipedia。“词嵌入是NLP中语言模型和表征技术的统称，概念上它是把一个维数为词库大小的高维空间嵌入维数较低的连续向量空间中，每个单词或词组被映射为实数域上的向量。” Word2Vec 是词嵌入技术的一种。

而词向量是词或词组在向量空间中的具体表示形式。

NNLM 通常可以看成三层网络：input-hidden/projection-output。Input 层填入词的 one-hot 表示，hidden/projection 层最初是 tanh 函数、Word2Vec 中简化为线性运算，output 层没争议，就是 Softmax 输出到词表。

通常，在语料上训练得到 NNLM 后，input-projection 层的权重作为 DNN 的 embedding layer 使用。此时输入的依然是 one-hot 。

考虑 one-hot 的运算特性不难看出，（根据表示方法不同）每个词对应的“词向量”就是权重矩阵/embedding layer 相应行/列的参数向量。

2 两条错误

前面说到，网上很多资料是有问题的。第一个错误在上文中已经指出，第二个错误与 one-hot 表示有关。这一点由苏剑林（https://spaces.ac.cn/archives/4122）指出。

很多资料指出/复制他人的观点认为 one-hot 的原罪在于其巨大二值稀疏矩阵效率低下。这种说法是有问题的。比如上文中苏剑林说的，词向量并没有提升计算效率，输入仍为 one-hot 形式，但多了嵌入层之后 one-hot 通过查表操作把长长的 0-1 表示变换到紧密的实数域向量上去了。

但苏剑林其实说的也不全对。另一半答案可以看这篇博客，依然是 Physcal 的：

http://www.cnblogs.com/neopenx/p/4570648.html

推荐本文读者多读几遍 Physcal 的两篇博文，作者的理解层次还是挺高的。

> 问题：维度过高
>
> 通常一个词库的大小是10^5，如果继续用二进制编码。那么一个句子的维度是10^5。
>
> 要知道，AlexNet的一张图片维度才256*256=65536, 就得拿GPU算好久，10^5基本得完蛋了。
>
> 实际上，10^5里，大部分都是维度都是废的，真正有用的特征就藏在那么几个维度中。
>
> 这说明，One-hot Representation表达的特征维度过高，需要降维。然而，这还不是最坑爹的缺陷。
>
> Bengio在2003年的A neural probabilistic language model中指出，维度过高，导致每次学习，都会强制改变大部分参数。
>
> 由此发生蝴蝶效应，本来很好的参数，可能就因为一个小小传播误差，就改的乱七八糟。
>
> 实际上，传统MLP网络就是犯了这个错误，1D全连接的神经元控制了太多参数，不利于学习到稀疏特征。
>
> CNN网络，2D全连接的神经元则控制了局部感受野，有利于解离出稀疏特征。
>
> 1.3中已经提过，在深度学习 NLP 任务中，文本依然以 one-hot 的形式输入网络。

解释得很清楚了。

one-hot 表示可以看作最粗糙但无损失的词表示方法，它的缺点并不是稀疏，而是无法表达词与词之间的关系，即所谓语义鸿沟。在大规模语料上训练 NNLM，可以利用 NN 学习到语料给出的语义信息、词间关系，从而克服实践中 one-hot 的缺点。

前两天跟同学聊天也提到了这个事情。NNLM 仅仅是词嵌入的一种工具，在实践中不见得哪都有效。他们在做的事情就是想办法用 CNN +贝叶斯直接在 one-hot 上训练网络。据说腾讯 AI 的 Boss 张潼之前做过这个工作。但根据上面的说法我对这种思路表示怀疑……

3 一点想法

3.1 学习体会

看皮果提的博文可以发现，虽然论文里基本没有理论的证明推导，但训练网络实际需要很多计算，数学都在 Hierarchical Softmax 和 Negative Sampling 的 BP 上了。想知道的话可以看他的博文和 Rong Xin 的 Word2Vec 参数解释，步骤详尽。

源码的编程技巧也不是我能讲的，事实上理解以上内容就已经十分吃力了。什么内存对齐、Huffman 树构建，完全搞不定。

Word2Vec 不是一切，虽然它几乎可以称为标配——速度快、省内存。此外 GloVe 和 FastText 也各有长短，后者同样以速度著称。

就 DR 这个问题来说，很多人认为一个最大的难点在于如何衡量词表示的好坏。对不同任务和数据集来说，各方法的结果表现差别很大，这里面水就深了。

有人好奇 Word2Vec 有没有其他实现版本。有。Tensorflow 官方有一个，但据说效率比 C 版差；Gensim 有一个基于 Python 的实现，用了 Cython + BLAS 加速，号称远超原版，但据说出来的结果跟原版不一样；国内有大佬搞了 HanLP，并加入了自己的 Java 实现，据说比原版快一倍但精度稍差（http://www.hankcs.com/nlp/word2vec.html）。

最后，有问题去读源码、读源码、读源码。

要说三遍。

3.2 word2vec 的缺陷（update）

上面说过，word2vec 通过精简原 NNLM 中的非线性提升效率，得到的结果非常不错，所谓的

$$vec(king)-vec(man)+vec(woman)\approx vec(queen)$$

说明训练出的 wordvec 之间竟然具有类似线性的关系。这是之前没有想到的。

通过大量类似的简化，Word2Vec 在保证还不错的准确度的前提下，具备了极高的效率。但问题也随之而来，简化是有代价的。Bengio 模型中，输入是窗口词拼接而来的，保留了先后次序，而 CBOW 和 Skip-Gram 对窗口内的词一视同仁，也就忽略了它们之间的次序关系。

另外，NNLM 本质是利用神经网络去搜索和记忆数据中的有效统计信息。比如 https://zhuanlan.zhihu.com/p/29364112 文提到的微博语料训练出的有害词模型，输入“垃圾”，与之相关度最高的依次是：

> 辣鸡
>
> 拉圾
>
> 腊鸡
>
> 狗屎
>
> 废物
>
> 渣子
>
> sb
>
> ...

表明这些词在原文中与“垃圾”的距离最近（在前在后），或者具有相近的含义。但这里的“含义”并不表示模型理解语义，而只是在训练中发现这些词在语句中是可以互相替换的——骂人的时候有用“垃圾”的，有用“辣鸡”的。这两点上文均有提及：相似的句子，相似位置/成分的词更接近；实际距离更近的词更接近。

这样的结果也是由网络的训练过程决定的。

转自： AI研习社