词嵌入来龙去脉 word embedding、word2vec

2017年08月14日 21:57:45 最小森林 阅读数：11412

0词嵌入来龙去脉

之前一段时间，在结合深度学习做NLP的时候一直有思考一些问题，

不少的terms like： 词向量、word embedding、分布式表示、word2vec、glove等等，

这一锅粥的名词术语分别代表什么，他们具体的关系是什么，他们是否处于平级关系？

整篇文章的构架是按照属于概念在逻辑上的先后大小顺序，一层一层一级一级地往下剖析、比较、说明。

另外说明下，here整篇文字内容相对是比较入门，甚至有的点可能描述的不太客观正确，限于当前的认知水平……还请您海涵，希望您在评论中指正！

1 NLP的核心关键：语言表示（Representation）

Deep Learning如何能在NLP中发挥出应有的real power呢？很明显，先不提如何设计出很强势的网络结构，不提如何在NLP中引入基于NN的解决例如情感分析、实体识别、机器翻译、文本生成这些高级任务，咱们首先得把语言表示这一关过了——如何让语言表示成为NN能够处理的数据类型。

我们看看图像和语音是怎么表示数据的：

在语音中，用音频频谱序列向量所构成的matrix作为前端输入喂给NN进行处理，good；在图像中，用图片的像素构成的matrix展平成vector后组成的vector序列喂给NN进行处理，good；那在自然语言处理中呢？噢你可能知道或者不知道，将每一个词用一个向量表示出来！想法是挺简单的，对，事实上就是这么简单，然而真有这么简单吗？可能没这么简单。

有人提到，图像、语音属于比较自然地低级数据表示形式，在图像和语音领域，最基本的数据是信号数据，我们可以通过一些距离度量，判断信号是否相似，在判断两幅图片是否相似时，只需通过观察图片本身就能给出回答。而语言作为人类在进化了几百万年所产生的一种高层的抽象的思维信息表达的工具，其具有高度抽象的特征，文本是符号数据，两个词只要字面不同，就难以刻画它们之间的联系，即使是“麦克风”和“话筒”这样的同义词，从字面上也难以看出这两者意思相同（语义鸿沟现象），可能并不是简单地一加一那么简单就能表示出来，而判断两个词是否相似时，还需要更多的背景知识才能做出回答。

那么据上是不是可以自信地下一个结论呢：如何有效地表示出语言句子是决定NN能发挥出强大拟合计算能力的关键前提！

2 NLP词的表示方法类型

接下来将按照上面的思路，引出各种词的表示方法。按照现今目前的发展，词的表示分为独热表示one-hot、分布式表示distributed。

2.1词的独热表示one-hot representation

NLP 中最直观，也是到目前为止最常用的词表示方法是 One-hot Representation，这种方法把每个词表示为一个很长的向量。这个向量的维度是词表大小，其中绝大多数元素为 0，只有一个维度的值为 1，这个维度就代表了当前的词。关于one-hot编码的资料很多，街货，这里简单举个栗子说明：

```
“话筒”表示为 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ...]
“麦克”表示为 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 ...]
```

每个词都是茫茫 0 海中的一个 1。这种 One-hot Representation 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。

现在我们分析他的不当处。1、向量的维度会随着句子的词的数量类型增大而增大；2、任意两个词之间都是孤立的，根本无法表示出在语义层面上词语词之间的相关信息，而这一点是致命的。

2.2词的分布式表示distributed representation

统的独热表示（ one-hot representation）仅仅将词符号化，不包含任何语义信息。如何将语义融入到词表示中？Harris 在 1954 年提出的分布假说（ distributional hypothesis）为这一设想提供了理论基础：上下文相似的词，其语义也相似。Firth 在 1957 年对分布假说进行了进一步阐述和明确：词的语义由其上下文决定（ a word is characterized by thecompany it keeps）。

到目前为止，基于分布假说的词表示方法，根据建模的不同，主要可以分为三类：基于矩阵的分布表示、基于聚类的分布表示和基于神经网络的分布表示。尽管这些不同的分布表示方法使用了不同的技术手段获取词表示，但由于这些方法均基于分布假说，它们的核心思想也都由两部分组成：一、选择一种方式描述上下文；二、选择一种模型刻画某个词（下文称“目标词”）与其上下文之间的关系。

3 NLP语言模型

在详细介绍词的分布式表示之前，需要将NLP中的一个关键概念描述清楚：语言模型。语言模型包括文法语言模型和统计语言模型。一般我们指的是统计语言模型。之所以要将语言模型摆在词表示方法之前，是因为后面的表示方法马上要用到这一概念。

统计语言模型： 统计语言模型把语言（词的序列）看作一个随机事件，并赋予相应的概率来描述其属于某种语言集合的可能性。给定一个词汇集合 V，对于一个由 V 中的词构成的序列S = 〈w1, · · · , wT 〉∈ Vn，统计语言模型赋予这个序列一个概率P(S)，来衡量S 符合自然语言的语法和语义规则的置信度。

用一句简单的话说，就语言模型就是计算一个句子的概率大小的这种模型。有什么意义呢？一个句子的打分概率越高，越说明他是更合乎人说出来的自然句子。

就是这么简单。常见的统计语言模型有N元文法模型（N-gram Model），最常见的是unigram model、bigram model、trigram model等等。形式化讲，统计语言模型的作用是为一个长度为 m 的字符串确定一个概率分布 P(w1; w2; :::; wm)，表示其存在的可能性，其中 w1 到 wm 依次表示这段文本中的各个词。一般在实际求解过程中，通常采用下式计算其概率值：

$$P(w_i \mid w_1, w_2, \ldots, w_{i-1}) \approx P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

同时通过这些方法均也可以保留住一定的词序信息，这样就能把一个词的上下文信息capture住。

具体的语言模型详情属于街货，详细请自行搜索。

4 词的分布式表示

4.1基于矩阵的分布表示

基于矩阵的分布表示通常又称为分布语义模型，在这种表示下，矩阵中的一行，就成为了对应词的表示，这种表示描述了该词的上下文的分布。由于分布假说认为上下文相似的词，其语义也相似，因此在这种表示下，两个词的语义相似度可以直接转化为两个向量的空间距离。

常见到的Global Vector 模型（ GloVe模型）是一种对“词-词”矩阵进行分解从而得到词表示的方法，属于基于矩阵的分布表示。

4.2基于聚类的分布表示

基于聚类的分布表示我也还不是太清楚，所以就不做具体描述。

4.3基于神经网络的分布表示，词嵌入（ word embedding）

==基于神经网络的分布表示一般称为词向量、词嵌入（word embedding）或分布式表示（distributed representation）==。这正是我们的主角today。

神经网络词向量表示技术通过神经网络技术对上下文，以及上下文与目标词之间的关系进行建模。由于神经网络较为灵活，这类方法的最大优势在于可以表示复杂的上下文。在前面基于矩阵的分布表示方法中，最常用的上下文是词。如果使用包含词序信息的 n-gram 作为上下文，当 n 增加时， n-gram 的总数会呈指数级增长，此时会遇到维数灾难问题。而神经网络在表示 n-gram 时，可以通过一些组合方式对 n 个词进行组合，参数个数仅以线性速度增长。有了这一优势，神经网络模型可以对更复杂的上下文进行建模，在词向量中包含更丰富的语义信息。

5 词嵌入（ word embedding）

5.1概念

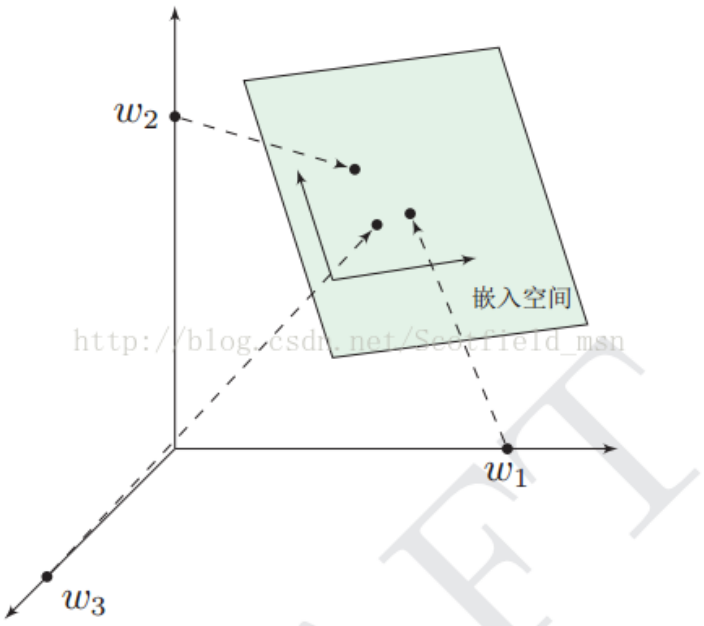
基于神经网络的分布表示又称为词向量、词嵌入，神经网络词向量模型与其它分布表示方法一样，均基于分布假说，核心依然是上下文的表示以及上下文与目标词之间的关系的建模。

前面提到过，为了选择一种模型刻画某个词（下文称“目标词”）与其上下文之间的关系，我们需要在词向量中capture到一个词的上下文信息。同时，上面我们恰巧提到了统计语言模型正好具有捕捉上下文信息的能力。那么构建上下文与目标词之间的关系，最自然的一种思路就是使用语言模型。从历史上看，早期的词向量只是神经网络语言模型的副产品。

2001年， Bengio 等人正式提出神经网络语言模型（ Neural Network Language Model ， NNLM），该模型在学习语言模型的同时，也得到了词向量。所以请注意一点：==词向量可以认为是神经网络训练语言模型的副产品==。

5.2理解

前面提过，one-hot表示法具有维度过大的缺点，那么现在将vector做一些改进：1、将vector每一个元素由整形改为浮点型，变为整个实数范围的表示；2、将原来稀疏的巨大维度压缩嵌入到一个更小维度的空间。如图示：



这也是词向量又名词嵌入的缘由了。

6 神经网络语言模型与word2vec

好了，到目前为止我们已经对的分布式表示以及词嵌入的概念的层级关系有了个理性的认识了，那这跟word2vec有什么联系?

6.1神经网络语言模型

上面说，通过神经网络训练语言模型可以得到词向量，那么，究竟有哪些类型的神经网络语言模型呢？个人所知，大致有这么些个：

```
a) Neural Network Language Model , NNLM
b) Log-Bilinear Language Model,  LBL
c) Recurrent Neural Network based Language Model, RNNLM
d) Collobert 和 Weston 在2008 年提出的 C&W 模型
e) Mikolov 等人提出了 CBOW（ Continuous Bagof-Words）和 Skip-gram 模型
```

到这，估计有人看到了两个熟悉的term：CBOW、skip-gram，有看过word2vec的同学应该对此有所了解。我们继续。

6.2word2vec与CBOW、Skip-gram

现在我们正式引出最火热的另一个term：word2vec。

上面提到的5个神经网络语言模型，只是个在逻辑概念上的东西，那么具体我们得通过设计将其实现出来，而实现CBOW（Continuous Bagof-Words）和 Skip-gram 语言模型的工具正是well-known word2vec！另外，C&W 模型的实现工具是SENNA。

所以说，分布式词向量并不是word2vec的作者发明的，他只是提出了一种更快更好的方式来训练语言模型罢了。分别是：连续词袋模型Continous Bag of Words Model(CBOW)和Skip-Gram Model，这两种都是可以训练出词向量的方法，再具体代码操作中可以只选择其一，不过据论文说CBOW要更快一些。

顺便说说这两个语言模型。统计语言模型statistical language model就是给你几个词，在这几个词出现的前提下来计算某个词出现的（事后）概率。CBOW也是统计语言模型的一种，顾名思义就是根据某个词前面的C个词或者前后C个连续的词，来计算某个词出现的概率。Skip-Gram Model相反，是根据某个词，然后分别计算它前后出现某几个词的各个概率。

以“我爱北京天安门”这句话为例。假设我们现在关注的词是“爱”，C＝2时它的上下文分别是“我”，“北京天安门”。CBOW模型就是把“我” “北京天安门” 的one hot表示方式作为输入，也就是C个1xV的向量，分别跟同一个VxN的大小的系数矩阵W1相乘得到C个1xN的隐藏层hidden layer，然后C个取平均所以只算一个隐藏层。这个过程也被称为线性激活函数(这也算激活函数？分明就是没有激活函数了)。然后再跟另一个NxV大小的系数矩阵W2相乘得到1xV的输出层，这个输出层每个元素代表的就是词库里每个词的事后概率。输出层需要跟ground truth也就是“爱”的one hot形式做比较计算loss。这里需要注意的就是V通常是一个很大的数比如几百万，计算起来相当费时间，除了“爱”那个位置的元素肯定要算在loss里面，word2vec就用基于huffman编码的Hierarchical softmax筛选掉了一部分不可能的词，然后又用nagetive samping再去掉了一些负样本的词所以时间复杂度就从O(V)变成了O(logV)。Skip gram训练过程类似，只不过输入输出刚好相反。

补充下，Word embedding的训练方法大致可以分为两类：一类是无监督或弱监督的预训练；一类是端对端（end to end）的有监督训练。无监督或弱监督的预训练以word2vec和auto-encoder为代表。这一类模型的特点是，不需要大量的人工标记样本就可以得到质量还不错的embedding向量。不过因为缺少了任务导向，可能和我们要解决的问题还有一定的距离。因此，我们往往会在得到预训练的embedding向量后，用少量人工标注的样本去fine-tune整个模型。

相比之下，端对端的有监督模型在最近几年里越来越受到人们的关注。与无监督模型相比，端对端的模型在结构上往往更加复杂。同时，也因为有着明确的任务导向，端对端模型学习到的embedding向量也往往更加准确。例如，通过一个embedding层和若干个卷积层连接而成的深度神经网络以实现对句子的情感分类，可以学习到语义更丰富的词向量表达。

6.3个人对word embedding的理解

现在，词向量既能够降低维度，又能够capture到当前词在本句子中上下文的信息（表现为前后距离关系），那么我们对其用来表示语言句子词语作为NN的输入是非常自信与满意的。

另外一点很实用的建议，在你做某一项具体的NLP任务时如你要用到词向量，那么我建议你：要么1、选择使用别人训练好的词向量，注意，得使用相同语料内容领域的词向量；要么2、自己训练自己的词向量。我建议是前者，因为……坑太多了。

说到这里，其实我并没有想继续说下去的打算了，即并没有打算将word2vec的数学原理、详解啥的统统来一顿讲了，因为我发现网上关于讲解word2vec的文章实在是太多了，多到几乎所有的文章都是一样的。所以我也没有必要再copy一份过来咯。

所以，要详细了解word2vec、cbow、skip-gram细节的请您仔细搜索。我相信，在了解了这一系列的前提上下文知识的背景下，你再去读word2vec相关的细节文章时，一定不会感到有多吃力