Doc2Vec的一个轻量级介绍

原创 ronghuaiyang AI公园 2019-11-04

作者：Gidi Shperber

编译：ronghuaiyang

导读

在这篇文章中，你将学习什么是**doc2vec**，它是如何构建的，它与**word2vec**有什么关系，你可以用它做什么，没有数学公式。

介绍

文本文档的数字表示是机器学习中的一个具有挑战性的任务。这种表示形式可以用于多种目的，例如：文档检索、web搜索、垃圾邮件过滤、主题建模等。

然而，没有很多好的技术可以做到这一点。许多任务使用众所周知的但过于简单的方法如词袋(BOW)，但结果将大多是平庸的，因为BOW丢掉了许多微妙的可能的良好的表示，比如考虑单词的顺序。

LDA也是一种常见的主题建模技术(从文本中提取主题/关键字)，但它很难调试，结果也很难评估。

在这篇文章中。我将回顾**doc2vec**的方法，在2014年由Mikilov和Le提出，我们要通过这篇文章提到很多次。值得一提的是，Mikilov也是word2vec的作者之一。

Doc2vec是一个非常好的技术。它很容易使用，结果很好，而且从它的名字就可以看出来，它主要基于word2vec。我们先来简单介绍一下word2vec。

word2vec

word2vec是一个众所周知的概念，用于从单词中生成表示向量。

网上有很多关于word2vec的好教程，但是如果描述**doc2vec**而没有**word2vec**，就没有意义了，所以我就简单介绍一下。

一般来说，当你喜欢使用单词构建模型时，简单地标记/one-hot编码是一种可行的方法。然而，当使用这种编码时，这些词就失去了它们的意义。比如，如果我们将*Paris*编码为*id_4*， *France*编码为*id_6*，*power*编码为*id_8*，那么*France*与*power*的关系将与*Paris*相同。我们希望*France*和*Paris*能比*France*和*power*更接近。

2013年在这篇文章：https://arxiv.org/abs/1301.3781中提出的**word2vec**，可以给你每个单词的数字表示，并且能够捕获上述关系。这是机器学习中一个更广泛概念的一部分——特征向量。

这种表示法封装了词与词之间的不同关系，如同义词、反义词或类似的东西，如这个：

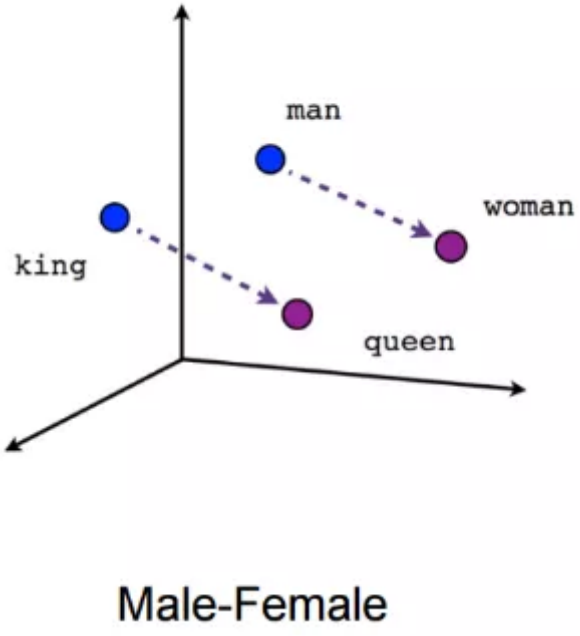

图1：国王对王后就像男人对女人。写关于word2vec不附加这个内容是非法的

Word2vec算法

这是怎么做到的呢？word2vec表示使用两种算法：连续的单词袋模型(**CBOW**)和**跳跃模型(** Skip-Gram)。

连续词袋模型

连续的单词包在当前单词周围创建一个滑动窗口，从“上下文” — 周围的单词来预测它。每个单词都表示为一个特征向量。经过训练，这些向量就变成了词向量。

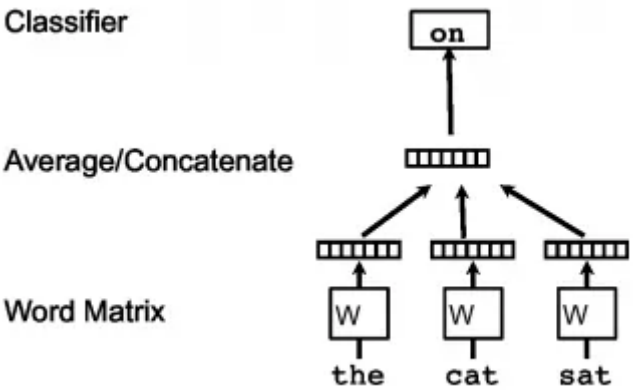

图2：CBOW算法示意图：用单词“the”，“cat”，“sat”来预测“on”

如前所述，表示相似单词的向量对于不同的距离度量是相近的，并且额外地封装了数值关系，如上面的*king-queen=man*。

Skip gram

第二种算法，在同一篇文章中有描述，与CBOW完全相反：我们不是每次预测一个单词，而是使用一个单词来预测所有周围的单词(“上下文”)。**Skip gram**比CBOW慢得多，但是对于不经常出现的单词，它被认为更准确。

Doc2vec

在理解了**word2vec**是什么之后，理解**doc2vec**是如何工作的就容易多了。

如前所述，**doc2vec**的目标是创建文档的数字表示，而不管其长度如何。但与单词不同的是，文档不是以单词这样的逻辑结构出现的，因此必须找到另一种方法。

Mikilov和Le使用的概念很简单，但很聪明：他们使用了**word2vec**模型，并添加了另一个向量(下面的段落ID)，如下所示：

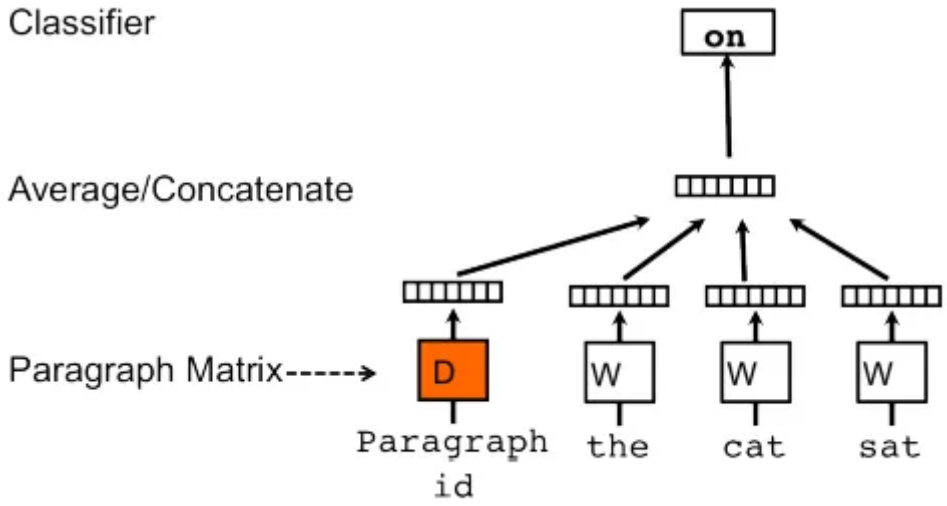

图3：PV-DM模型

如果你对上面的示意图感到很熟悉，那是因为它是CBOW模型的一个小扩展。但是，除了使用单词来预测下一个单词之外，我们还添加了另一个特征向量，它对于每个文档是唯一的。

因此，当训练单词向量W时，也训练了文档向量D，在训练结束时，它就有了文档的数字表示。

上面的模型称为*Distributed Memory version of Paragraph Vector*(PV-DM)。它就像一个记忆体，记住当前上下文缺少的内容 — 或者作为段落的主题。单词向量表示单词的概念，而文档向量表示文档的概念。

在word2vec中，可以使用另一种类似于skip-gram的算法，即**Distributed Bag of Words version of Paragraph Vector** (PV-DBOW)。

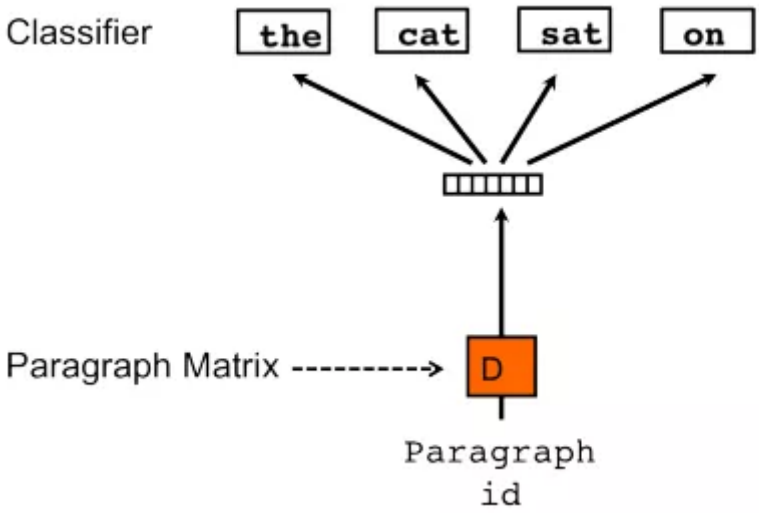

图4：PV-DBOW模型

在这里，这个算法实际上更快(与**word2vec**相反)，并且消耗更少的内存，因为不需要保存词向量。

在这篇文章中，作者声明他们推荐使用这两种算法的组合，尽管**PV-DM**模型更优，并且通常会自己就可以得到最先进的结果。

doc2vec模型可按以下方式使用：对于训练，需要一组文档。每个单词生成一个单词向量**W**，每个文档生成一个文档向量**D**。该模型还为softmax隐层训练权重。在推理阶段，可以使用一个新的文档，然后固定所有的权值来计算文档向量。

模型评估和一点想法

这种无监督模型的问题在于，它们没有被训练去完成它们本来要完成的任务。比如说， **word2vec**训练完成语料库中的包围词，但用于估计词之间的相似度或关系。因此，衡量这些算法的性能可能具有挑战性。我们已经看到了“国王”、“皇后”、“男人”、“女人”的例子，但我们想让它成为一种评估机器学习模型的严格方法。

因此，在训练这些算法时，我们应该注意相关的度量。**word2vec**的一个可能的度量标准是对上述示例的概括，称为类比推理。它包含许多类似的组合，如下：

- *happy happily — furious furiously*
- *immediate immediately — infrequent infrequently*
- *slowing slowed — sleeping slept*
- *spending spent — striking struck*

这个任务的成功之处在于，当计算匹配对之间的距离时，可以得到非常接近的结果。

数据集在http://download.tensorflow.org/data/questions-words.txt。

Doc2vec在文章中测试了两个任务：第一个是**情绪分析**，第二个类似于上面的类比推理。

这是文章中的三段。这些段落的数据集被用来比较模型。很容易看出哪两个比较接近：

- **Paragraph 1:** calls from (000) 000 - 0000 . 3913 calls reported from this number . according to 4 reports the identity of this caller is american airlines .

- **Paragraph 2:** do you want to find out who called you from +1 000 - 000 - 0000 , +1 0000000000 or (000) 000 - 0000 ? see reports and share information you have about this caller

- **Paragraph 3:** allina health clinic patients for your convenience , you can pay your allina health clinic bill online . pay your clinic bill now , question and answers...

这个数据集(据我所知没有共享)用来比较一些模型，**doc2vec**是最好的：

Model	Error rate
Vector Averaging	10.25%
Bag-of-words	8.10 %
Bag-of-bigrams	7.28 %
Weighted Bag-of-bigrams	5.67%
Paragraph Vector	**3.82%**

现实中的挑战 - ScaleAbout

我的一个客户，使用机器学习方法来进行you-tube视频到内容文章的匹配。Doc2vec似乎是一个很好的匹配方法。

有个例子是这样的，有一篇文章，是关于在家里用树桩做灯的，在文章的底部，可以看到4部木工相关的视频。

ScaleAbout当前的模型使用标签机制对视频和文章进行标注(“topic modeling”)，并测量标签之间的距离。

ScaleAbout有一些与客户主题相关的语料库。比如说，有一个10万手动标记的文件“do it yourself”，就像上面说过的，是给出版商准备的。每篇文章有17个可能的标签。如“家居装饰”、“园艺”、“改建及翻新”等。在这个实验中，我们决定尝试使用doc2vec和其他一些模型来预测标签。

ScaleAbout目前最好的模型是一个卷积神经网络，它建立在**word2vec**的基础上，在预测文档标签方面达到了**70%**的准确率。

Doc2vec模型本身是一个无监督的方法，所以需要稍微调整一下“参与”这个比赛。幸运的是，在大多数情况下，我们可以使用一些技巧：如果你还记得，在图3中我们添加了另一个文档向量，它对于每个文档都是惟一的。如果你想一下，可以添加更多的向量，它们不一定是唯一的：例如，如果我们的文档有标签(实际上我们有)，我们可以添加它们，并得到它们作为向量的表示。

此外，它们不必是唯一的。通过这种方式，我们可以将17个标记中的一个添加到唯一的文档标记中，并为它们创建一个doc2vec表示！见下图：

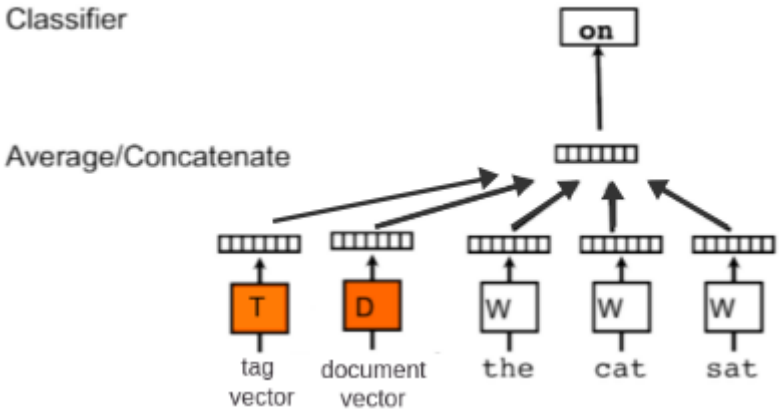

图5：带标签向量的doc2vec模型

我们使用**gensim**实现了doc2vec。下面是gensim TaggedDocument：

```
In [168]: tagged_docs[3]
Out[168]: TaggedDocument(words=['aftershere', 'project', 'finishing', 'stages', 'home', 'decor', 'kitchen', 'design', 'beforeher
e', 'project', 'finishing', 'stages', 'home', 'decor', 'kitchen', 'design', 'afterhere', 'project', 'finishing', 'stag
es', 'home', 'decor', 'kitchen', 'design', 'beforehere', 'project', 'finishing', 'stages', 'home', 'decor', 'kitchen',
 'design'], tags=['Remodeling & Renovating', 'SENT_3'])
```

gensim TaggedDocument。SENT_3是惟一的文档id，remodeling和renovating是标记

使用**gensim** doc2vec非常简单。像往常一样，模型应该被初始化，训练几个阶段：

```
In [134]: model = gensim.models.Doc2Vec(tagged_docs, dm = 0, alpha=0.025, size= 20, min_alpha=0.025, min_count=0)
          # 10K docs =15sec on SAinteg

In [163]: for epoch in range(10):
              if epoch % 2 == 0:
                  print ('Now training epoch %s'%epoch)
              model.train(tagged_docs_small,total_examples=model.corpus_count)
              model.alpha -= 0.002  # decrease the learning rate
              model.min_alpha = model.alpha  # fix the learning rate, no decay

          Now training epoch 0
          Now training epoch 2
          Now training epoch 4
          Now training epoch 6
          Now training epoch 8
```

然后我们可以检查每个唯一的**文档**与每个**标签**的相似度，这样做：

```
In [172]: model.docvecs.similarity('Gardening & Landscaping','SENT_3')
Out[172]: 0.23512461864743456
```

预测与文档相似度最高的标签。

使用这种方法，我们在100K篇文章中只训练了10K篇，我们的准确率就达到了74%，比以前更好。

总结

我们已经看到，通过一些调整，我们可以从一个已经非常有用的**word2vec**模型中获得更多。这很好，因为正如前面所说，在我看来，表示文档的标记和匹配还有很长的路要走。

此外，这表明，这是一个很好的例子，说明机器学习模型如何封装了更多的能力，而不仅仅是它们所训练的特定任务。这可以在深度CNNs中看到，它训练用于对象分类，但也可以用于语义分割或聚类图像。

最后，如果你有一些与文档相关的任务 — 这可能是一个很好的模型！