研究方向

- 搜索引擎与自然语言处理
- 文本挖掘与机器学习
- 情感分析与观点挖掘
- 面向生物医学领域的文本挖掘

学术报告

- 06-05 罗凌 神经网络结构在命名
- 06-02 徐博 在CQA检索中建模未匹
- 04-13 李虹磊 基于迁移学习和深
- 04-07 刘晓霞 从点嵌入到团嵌入
- 03-31 彭钰莹-在临床医学领域通
- 03-17 钱凌飞 -基于LSTM的神经网

资源下载

- 情感本体库 NEW
- 学习资料
- 学术论文
- 软件工具
- 其他下载

杨阳 word2vec使用指导

新闻来源：IR实验室 发布时间：2013/11/21 17:47:46

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。

一、理论概述

（主要来源于http://licstar.net/archives/328这篇博客）

1. 词向量是什么

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。

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。

举个栗子，

“话筒”表示为 [0 0 0 1 0 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 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。

当然这种表示方法也存在一个重要的问题就是“词汇鸿沟”现象：任意两个词之间都是孤立的。光从这两个向量中看不出两个词是否有关系，哪怕是话筒和麦克这样的同义词也不能幸免于难。

Deep Learning 中一般用到的词向量并不是刚才提到的用 One-hot Representation 表示的那种很长很长的词向量，而是用 Distributed Representation（不知道这个应该怎么翻译，因为还存在一种叫“Distributional Representation”(类似，LDA中用topic表示词语的词向量的表示方法）表示的一种低维实数向量。这种向量一般是这个样子：[0.792, -0.177, -0.107, 0.109, -0.542, ...]。维度以 50 维和 100 维比较常见。

2. 词向量的来历

Distributed representation 最早是 Hinton 在 1986 年的论文《Learning distributed representations of concepts》中提出的。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。

3. 词向量的训练

要介绍词向量是怎么训练得到的，就不得不提到语言模型。到目前为止我了解到的所有训练方法都是在训练语言模型的同时，顺便得到词向量的。

这也比较容易理解，要从一段无标注的自然文本中学习出一些东西，无非就是统计出词频、词的共现、词的搭配之类的信息。而要从自然文本中统计并建立一个语言模型，无疑是要求最为精确的一个任务（也不排除以后有人创造出更好更有用的方法）。既然构建语言模型这一任务要求这么高，其中必然也需要对语言进行更精细的统计和分析，同时也会需要更好的模型，更大的数据来支撑。目前最好的词向量都来自于此，也就不难理解了。

词向量的训练最经典的有 3 个工作，C&W 2008、M&H 2008、Mikolov 2010。当然在说这些工作之前，不得不介绍一下这一系列中 Bengio 的经典之作

4. 词向量的评价

词向量的评价大体上可以分成两种方式，第一种是把词向量融入现有系统中，看对系统性能的提升；第二种是直接从语言学的角度对词向量进行分析，如相似度、语义偏移等。

4.1 提升现有系统

词向量的用法最常见的有两种：

1. 直接用于神经网络模型的输入层。如 C&W 的 SENNA 系统中，将训练好的词向量作为输入，用前馈网络和卷积网络完成了词性标注、语义角色标注等一系列任务。再如 Socher 将词向量作为输入，用递归神经网络完成了句法分析、情感分析等多项任务。

2. 作为辅助特征扩充现有模型。如 Turian 将词向量作为额外的特征加入到接近 state of the art 的方法中，进一步提高了命名实体识别和短语识别的效果。

4.2 语言学评价

还有一个有意思的分析是 Mikolov 在 2013 年刚刚发表的一项发现。他发现两个词向量之间的关系，可以直接从这两个向量的差里体现出来。向量的差就是数学上的定义，直接逐位相减。比如 C(king)−C(queen)≈C(man)−C(woman)。更强大的是，与 C(king)−C(man)+C(woman) 最接近的向量就是 C(queen)。

为了分析词向量的这个特点， Mikolov 使用类比（analogy）的方式来评测。如已知 a 之于 b 犹如 c 之于 d。现在给出 a、b、c，看 C(a)−C(b)+C(c) 最接近的词是否是 d。

在文章 Mikolov 对比了词法关系（名词单复数 good-better:rough-rougher、动词第三人称单数、形容词比较级最高级等）和语义关系（clothing-shirt:dish-bowl）

这些实验结果中最容易理解的是：语料越大，词向量就越好。其它的实验由于缺乏严格控制条件进行对比，谈不上哪个更好哪个更差。不过这里的两个语言学分析都非常有意思，尤其是向量之间存在这种线性平移的关系，可能会是词向量发展的一个突破口。

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

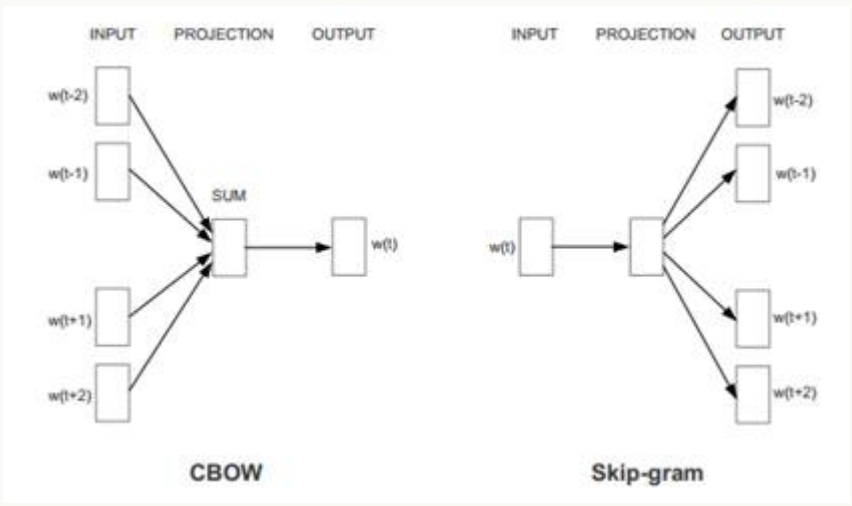
Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

关于Deep Lerning In Nlp的一些相关论文，《Deep Learning in NLP （一）词向量和语言模型》（http://licstar.net/archives/328）这篇博客总结的非常的好。以上内容大多数都是截取原博客内容。

二、实际操作

这篇文章是最近几天看word2vec源码以及相关神经网络训练词向量论文之后的个人小小的总结，主要是针对word2vec的使用，做一下介绍。望大家使用的过程中，少走弯路。

word2vec工具中包含了对两种模型的训练，如下图。在训练每种模型的时候又分HS和NEG两种方法。(看图就可以发现，其实word2vec并不deep……)



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,但通过阅读其源码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。下面列举一下目前出现的版本以及相对应的地址，供大家选择。如下表：

版本	地址	CBOW		Skip-Gram	
C	http://word2vec.googlecode.com/svn/trunk/	HS	NEG	HS	NEG
python	http://radimrehurek.com/gensim/			HS	
Java	https://github.com/ansjsun/Word2VEC_java	HS		HS	
C++	https://github.com/jdeng/word2vec	未知	未知	未知	未知

以上代码，C++版本的我没有看过。最权威的当然是C语言版本，但是阅读起来比较困难一点。Python版本有优化处理，所以速度相对来说也不慢，但只是实现了分层softmax方法对skip-gram模型进行训练。Java版本分别实现了分层softmax方法对CBOW模型和skip-gram模型进行训练。C++版本的没有阅读其代码，所以未知……

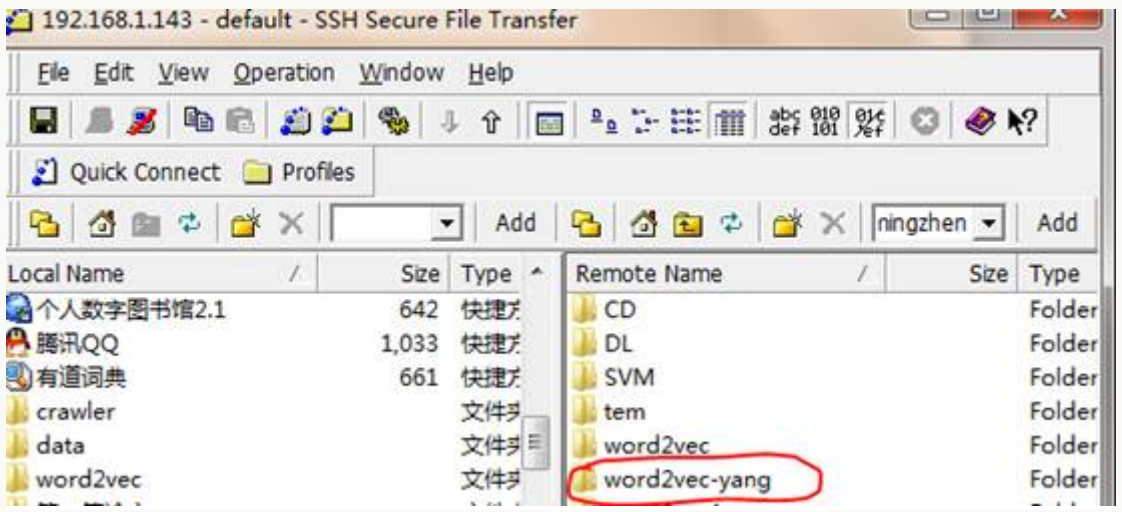
使用之前，先贴一些论文中对两个模型和不同方法的评价图片，方便大家根据不同任务进行不同训练。

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

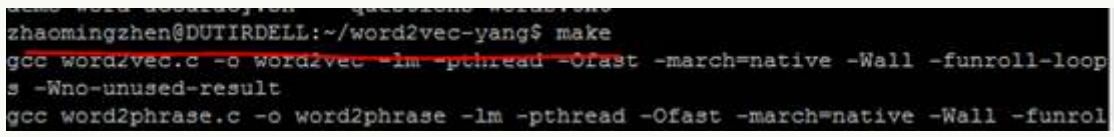
Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	**61**
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53
The following results use 10^{-5} subsampling				
NEG-5	14	61	58	60
NEG-15	36	61	61	**61**
HS-Huffman	21	52	59	55

下面以c语言正式版本为例，来介绍word2vec的使用。

首先我们将google word2vec项目源码checkout 到本机，具体地址是http://word2vec.googlecode.com/svn/trunk/ 使用ssh登录实验室Linux服务器，地址192.168.1.143。将刚才checkout的文件，上传到服务器中。



进入目录，命令行输入make指令，进行编译。



这样我们就可以开始使用，word2vec工具了。

1.将文本语料进行分词，以空格,tab隔开都可以，中文分词工具可以使用张华平博士的NLPIR2013 http://ictclas.nlpir.org/ 喜欢用Python 的童鞋也可以使用结巴分词https://github.com/fxsjy/jieba。

2.将分好词的训练语料进行训练，假定我语料名称为test.txt且在word2vec目录中。输入命令：

./word2vec -train test.txt -output vectors.bin -cbow 0 -size 200 -window 5 -negative 0 -hs 1 -sample 1e-3 -threads 12 -binary 1

以上命令表示的是输入文件是test.txt，输出文件是vectors.bin，不使用cbow模型，默认为Skip-Gram模型。 每个单词的向量维度是200，训练的窗口大小为5就是考虑一个词前五个和后五个词语（实际代码中还有一个随机选窗口的过程，窗口大小<=5）。不使用NEG方法，使用HS方法。-sampe指的是采样的阈值，如果一个词语在训练样本中出现的频率越大，那么就越会被采样。-binary为1指的是结果二进制存储，为0是普通存储（普通存储的时候是可以打开看到词语和对应的向量的）除了以上命令中的参数，word2vec还有几个参数对我们比较有用比如-alpha设置学习速率，默认的为0.025. -min-count设置最低频率，默认是5，如果一个词语在文档中出现的次数小于5，那么就会丢弃。-classes设置聚类个数，看了一下源码用的是k-means聚类的方法。

- 架构：skip-gram（慢、对罕见字有利）vs CBOW（快）
- 训练算法：分层softmax（对罕见字有利）vs 负采样（对常见词和低纬向量有利）
- 欠采样频繁词：可以提高结果的准确性和速度（适用范围1e-3到1e-5）
- 文本（window）大小：skip-gram通常在10附近，CBOW通常在5附近

3.训练好模型之后，得到vectors.bin这个模型文件。vectors.bin这个文件就是文档中词语和其对应的向量，这个向量的维度是你训练时设置的参数大小。下面我们可以利用这个model做很多自然语言处理的任务了。Google代码里面提供了

distance的一个应用，说白了就是读取模型文件中每一个词和其对应的向量，计算所输入query的词，与其他所有词语的cosine相似度，排序，返回结果。同理训练命令中的-classes参数，也是获取每个词对应的向量之后，对词语进行k-means聚类。对于训练出来的模型进行操作，我推荐大家使用http://blog.csdn.net/zhaoxinfan/article/details/11640573这个java版本的模型读取类，比较方便，读取模型之后大家就可以来实现各种各样有趣的东西了。下面举几个例子：

a.计算两个词语相似度，如图1计算asp与net的相似度为0.6215127

```
import java.io.BufferedInputStream;

public class Word2VEC {

    public static void main(String[] args) throws IOException {
        Word2VEC vec = new Word2VEC();
        vec.loadModel("data/vectors.bin");
        float result = vec.similarity("asp", "net");
        System.out.println(result);
```

Console | Error Log | PyUnit

```
<terminated> Word2VEC [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (2013-11-21 上午10:39:33)
0.6215127
```

图1

b.列出所有相似词语列表,如图2为“php”的结果。

```
        System.out.println("One word analysis");
        Set<WordEntry> result = new TreeSet<WordEntry>();
        result = vec.distance("php");


        Iterator iter = result.iterator();
        while (iter.hasNext()) {
            WordEntry word = (WordEntry) iter.next();
```

Console | Error Log | PyUnit

```
<terminated> Word2VEC [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (20
One word analysis
asp 0.79655695
程序员 0.78699213
java 0.7489228
软件开发 0.7382306
mysql 0.69717664
linux 0.6641003
zend 0.66255957
程序开发 0.6624413
thinkphp 0.6566815
后台程序 0.6509557
phpchina 0.6490292
欣才 0.64681137
创恩 0.64134043
divcss 0.63653207
文豆 0.63471615
javascript 0.63305235
jsp 0.62776184
多谪 0.6234954
开发 0.62214094
前端开发 0.62212
mssql 0.61484003
net 0.61243045
```

图二

c.寻找对应关系：如图3： 男人-男孩 女人-？ 如图4：内蒙-呼和浩特 河北-？

```
        System.out.println("*******************************");
        System.out.println("Three word analysis");
        TreeSet<WordEntry> result = vec.analogy("男人", "女人", "男孩");
        //男人-男孩 女人-?
        Iterator iter = result.iterator();
        while (iter.hasNext()) {
            WordEntry word = (WordEntry) iter.next();
            System.out.println(word.name + " " + word.score);
        }
    }

    /**
     * 计算word1和word2的相似度
```

Console | Error Log | PyUnit

```
<terminated> Word2VEC [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (2013-11-21 上午10:52:56)
*******************************
Three word analysis
女孩 0.89474124
小女孩 0.72787994
潮妈 0.7164343
男送 0.71036786
十二岁 0.70887923
君秀 0.7020763
一周岁 0.70164496
矮个子 0.6994891
穿多大 0.69902754
应穿 0.6980718
能涂 0.695152
```

图3

```
37        System.out.println("******************************");
38        System.out.println("Three word analysis");
39  //    TreeSet<WordEntry> result = vec.analogy("男人", "女人", "男孩");
40  //    //男人-男孩 女人-?
41        TreeSet<WordEntry> result = vec.analogy("内蒙", "河北", "呼和浩特");
42        //男人-年 女人-?
43        Iterator iter = result.iterator();
44        while (iter.hasNext()) {
45            WordEntry word = (WordEntry) iter.next();
46            System.out.println(word.name + " " + word.score);
47        }
48    }
49
```

Console | Error Log | PyUnit

```
<terminated> Word2VEC [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (2013-11-21 上午10:57:51)
******************************
Three word analysis
石家庄 0.7702285
唐山 0.6858602
河北省 0.68007386
保定 0.67245364
沧州 0.65414995
廊坊 0.6321514
裕华区 0.63170195
桥东区 0.6029325
桥西区 0.5946565
保定市 0.5855945
长安区 0.5843909
```

图4

d. 删选不合群的词语，可以用来做特征选择：如图5所示：

```
49        HashSet<String> words = new HashSet<String>();
50        words.add("大连");
51        words.add("asp");
52        words.add("培训");
53        words.add("java");
54        words.add("面包");
55        words.add("教育");
56        words.add("c");
57        words.add("php");
58        words.add("jsp");
59
60        List<String> word_list = vec.doesnt_match(words);
61        for(String word:word_list){
62            System.out.println(word);
63        }
64    }
```

Console | Error Log | PyUnit

```
<terminated> Word2VEC [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (2013-11-21 上午11:27:49)
教育
面包
大连
```

图5