CS224n笔记——词向量评价与再训练

原创 管冲 浪漫主义AI 2018-01-24

目前我们已经学习了一些训练词向量的方法，现在让我们看看如何去评价词向量的质量。

评价方式分为内部评价和外部评价。

内部评价是我们在词向量训练技术中的具体的中间子任务上的评价，比如类比,这些子任务通常毕竟简单快速，能够帮助我们理解词向量，对于词向量的性能有具体的数值指标。这样做主要是因为一个具体的机器学习任务通常需要很多的时间和计算资源来完成，且词向量的性质会受到机器学习模型的好坏的干扰，所以有时候我们没有必要去进行一个具体的任务。

外部评价就是在真实任务上的评价，词向量的评价依赖于任务的结果，毕竟我们训练词向量是为了完成具体任务，所以这也是必不可少的，并且很多时候我们需要根据具体的任务来选择合适的词向量训练方法。然而当某个具体任务的模型表现很差时，我们常常并不能弄清楚到底是那部分有问题，所以还需要内部评价的支撑。

内部评价：

词向量类比是一种比较流行的词向量内部评价方式。在此任务中，我们的目标是解决这个问题：

$$a : b : : c : ?$$

也就是，哪个词与c的关系跟a与b的关系很像？

具体的数学表达为：

$$d = \underset{i}{\operatorname{argmax}} \frac{(x_b - x_a + x_c)^T x_i}{\|x_b - x_a + x_c\|}$$

这个式子的来源是xb-xa=xd-xc，亦即xd=xb-xa+xc。

在使用类比评价时要非常小心，类比条件的选取要仔细考虑。看一下下面这个例子：

City 1 : State containing City 1 : : City 2 : State containing City

Input	Result Produced
Chicago : Illinois : : Houston	Texas
Chicago : Illinois : : Philadelphia	Pennsylvania
Chicago : Illinois : : Phoenix	Arizona
Chicago : Illinois : : Dallas	Texas
Chicago : Illinois : : Jacksonville	Florida
Chicago : Illinois : : Indianapolis	Indiana
Chicago : Illinois : : Austin	Texas
Chicago : Illinois : : Detroit	Michigan
Chicago : Illinois : : Memphis	Tennessee
Chicago : Illinois : : Boston	Massachusetts

这是同一个国家的两个城市的类比，我们发现类比的结果比较糟糕。原因很明显，我们选取的这个类比问题的答案太多了！甚至于还有很多同名的美国城市、城镇或者乡村。如果是下面这个类比问题：

Capital City 1 : Country 1 : : Capital City 2 : Country 2

你需要注意的是你的数据集的时效，因为首都城市可能会有变化。

我们不仅可以在语义上做类比，还能在语法上做类比：

Input	Result Produced
bad : worst : : big	biggest
bad : worst : : bright	brightest
bad : worst : : cold	coldest
bad : worst : : cool	coolest
bad : worst : : dark	darkest
bad : worst : : easy	easiest
bad : worst : : fast	fastest
bad : worst : : good	best
bad : worst : : great	greatest

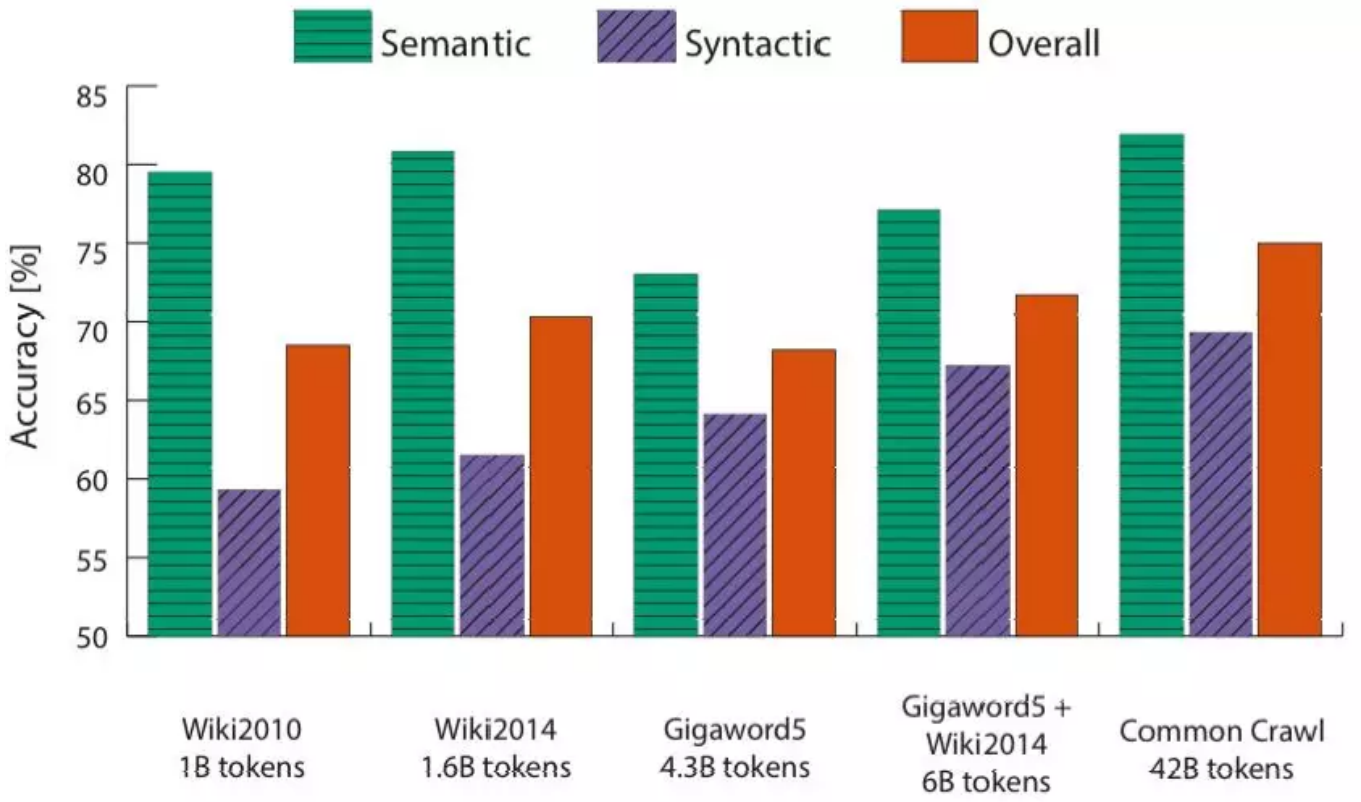
x下面我们在不同大小的数据集上做不同模型不同向量维度的评估：

Model	Dimension	Size	Semantics	Syntax	Total
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVE	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW	300	6B	63.6	67.4	65.7
SG	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	81.9	69.3	75.0

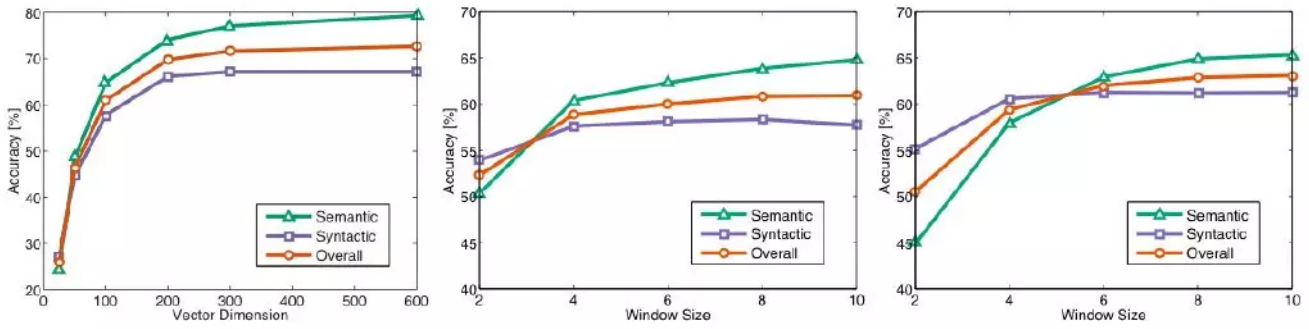
观测到三个主要的结论：

1. 词向量的性能非常依赖训练模型的选取，毕竟不同的模型能利用上的数据信息都不一样。
2. 数据集越大，性能越好，这一点和机器学习的普遍规律是相同的。
3. 向量维度过大或过小都会导致性能的下降，这一点可以从机器学习普遍规律推测得到，维度过小就不能很好地捕获数据集上的各种信息，也就是高偏差，underfitting；维度过大，就会捕获过多的数据上的噪声，泛化能力变差，也就是高方差，overfiiting。

   下图表现了数据集对性能的影响：



下图表现了向量维度和窗口大小的影响：



(a) Symmetric context (b) Symmetric context (c) Asymmetric context

除类比法之外，我们还可以使用人工标注的词性关系数据来验证我们的词向量的性能。这种方式人力成本很高，但是可操控性强。

外部评估：

外部评估衡量的是词向量模型对一个具体任务的影响。在外部评估中有一个隐含的假设，那就是对词向量的质量有一个一致的、全局的排名，高质量的词向量必然会改善下游任务的性能。但是，这个和假设并不成立：不同的任务需要不同的词向量。所以外部评估仅仅用来比较词向量之间的相对优劣势，而不能当作词向量质量的广泛意义上的评估。

首先是一个名词短语分块任务，注重语法上的分类：

	dev	test	p-value
Baseline	94.18	93.78	0.000
Rand. Proj.	94.33	93.90	0.006
GloVe	94.28	93.93	0.015
H-PCA	94.48	93.96	0.029
C&W	**94.53**	**94.12**	
CBOW	94.32	93.93	0.012
TSCCA	**94.53**	94.09	0.357

Table 4: F1 chunking results using different word embeddings as features. The p-values are with respect to the best performing method.

后是情感分析任务，注重语义分析：

	test	p-value
BOW (baseline)	88.90	$7.45 \cdot 10^{-14}$
Rand. Proj.	62.95	$7.47 \cdot 10^{-12}$
GloVe	74.87	$5.00 \cdot 10^{-2}$
H-PCA	69.45	$6.06 \cdot 10^{-11}$
C&W	72.37	$1.29 \cdot 10^{-7}$
CBOW	**75.78**	
TSCCA	75.02	$7.28 \cdot 10^{-4}$

Table 5: F1 sentiment analysis results using different word embeddings as features. The p-values are with respect to the best performing embedding.

中p-value是在句子级应用随机化的计算结果。

这两张图细节太多，暂且不论，不过至少说明了一个观点，那就是至少目前没有全局最优的词向量模型，最优的处理方式就是根据具体的任务选取最优的词向量模型。

外部任务训练：

多数NLP任务都可以认定为分类任务，典型的有情感分析和命名实体判别。对于分类任务，我们设想这样的一个数据集：

$$\{x^{(i)}, y^{(i)}\}_1^N$$

x为d维词向量，y是c维的one-hot向量标注。

在其他机器学习任务中，我们都是给定输出和标注，训练模型的参数，但是，在NLP任务中，我们还可以有这样一种训练思路：对输入词向量进行再训练。

在经过内部评估后，我们选取了比较合适的训练策略得到了较好的词向量，在很多情况下，这样的词向量已经可以在外部任务中取得很好的效果，但是，也有可能在外部任务中再训练以获得更好的性能。不过，再训练词向量是有风险的。

如果想要再训练词向量，首先要保证数据集足够大，能覆盖到词库中的大部分词，这是因为我们使用word2vec或者glove训练词向量时，所有的词向量都在同一个词空间中，而如果在一个较小的数据集上再训练，那么使得经过再训练的词向量发生了空间的偏移，导致没有训练的词向量与之存在于不同的词空间，这样自然会使得性能下降。所以，小数据集上并不能使用再训练。

接下来让我们考虑一个softmax分类任务。

使用softmax函数计算分类概率：

$$p(y_j = 1|x) = \frac{\exp(W_j.x)}{\sum_{c=1}^{C} \exp(W_c.x)}$$

使用cross-entropy函数计算损失函数：

$$-\sum_{j=1}^{C} y_j \log(p(y_j = 1|x)) = -\sum_{j=1}^{C} y_j \log\left(\frac{\exp(W_j.x)}{\sum_{c=1}^{C} \exp(W_c.x)}\right)$$

假设k(i)是样本i中one-hot向量y中值为1的维度的索引，那么在N个样本点上的损失函数为：

$$-\sum_{i=1}^{N} \log\left(\frac{\exp(W_{k(i)}.x^{(i)})}{\sum_{c=1}^{C} \exp(W_c.x^{(i)})}\right)$$

如果我们要进行再训练，那么在训练过程中我们更新的参数会非常多，也就非常容易过拟合，所以需要添加一个正则化项：

$$-\sum_{i=1}^{N}\log\left(\frac{\exp(W_{k(i)}.x^{(i)})}{\sum_{c=1}^{C}\exp(W_c.x^{(i)})}\right)+\lambda\sum_{k=1}^{C\cdot d+|V|\cdot d}\theta_k^2$$

窗口分类：

在实际任务中，使用单个的词向量进行分类是很困难的，因为人类语言的单词往往蕴含着多重含义，需要根据上下文进行语义判别，因此，模型的输入一般是词向量的序列，这个序列包括中心词和它的上下文单词。上下文的单词数成为窗口大小，其选择取决于任务的性质，一般语法任务会选取较小的窗口，语义任务会选取较大的窗口。

在上面的sotfmax分类任务中，我们可以把输入词向量变成：

$$x_{window}^{(i)}=\begin{bmatrix}x^{(i-2)}\\x^{(i-1)}\\x^{(i)}\\x^{(i+1)}\\x^{(i+2)}\end{bmatrix}$$

同时，在训练中更新词向量时，也会更新整个窗口中的词向量：

$$\delta_{window}=\begin{bmatrix}\nabla_{x^{(i-2)}}\\\nabla_{x^{(i-1)}}\\\nabla_{x^{(i)}}\\\nabla_{x^{(i+1)}}\\\nabla_{x^{(i+2)}}\end{bmatrix}$$