02 Word Vectors 2 and Word Window Classification 笔记

1. GloVe

GloVe的全称叫Global Vectors for Word Representation，它是一个基于**全局词频统计**（count-based & overall statistics）的词表征（word representation）工具，它可以把一个单词表达成一个由实数组成的向量，这些向量捕捉到了单词之间一些语义特性，比如相似性（similarity）、类比性（analogy）等。我们通过对向量的运算，比如欧几里得距离或者cosine相似度，可以计算出两个单词之间的语义相似性。

1.1 之前的词向量方法

词向量的两种方式：

- **基于统计**(LSA)

  通过矩阵分解得到词向量
  有效地利用全局统计信息，但主要用于捕获单词的相似性，在单词类比等任务中表现不佳

- **shallow window-based**(即skip-gram和cbow)

  通过在上下文中预测来学习词向量
  在语言相似性之外捕获复杂语言模式，但不擅长利于全局共现统计信息

1.2 共现矩阵

1.2.1 共现矩阵符号

X 表示词与词之间的共现矩阵

X_{ij} 表示词 j 出现在词 i 上下文的次数

$X_i = \sum_k X_{ik}$ 表示任意词 k 出现在词上下文 i 的次数

$P_{ij} = P(w_j|w_i) = \frac{X_{ij}}{X_i}$ 表示词 j 出现在词 i 上下文的概率

注：个人理解此条件概率公式是定义式，下文将出现条件概率的计算式，这里的定义式和条件式的概念来自于高中物理

1.2.2 计算复杂度分析

计算这个共现矩阵需要遍历一次整个语料库来来进行统计，对于大型语料库，这种遍历的代价很高，但这是一次就结束的代价。

1.3 原理实现

1.3.1 构造共现矩阵

根据语料库构建一个共现矩阵X，矩阵中的每一个元素 X_{ij} 代表单词i和上下文单词j在特定大小的上下文窗口（context window）内共同出现的次数。一般而言，这个次数的最小单位是1，但是GloVe不这么认为：它根据两个单词在上下文窗口的距离d，提出了一个衰减函数（decreasing weighting）：decay=1/d用于计算权重，也就是说距离越远的两个单词所占总计数（total count）的权重越小。

> In all cases we use a decreasing weighting function, so that word pairs that are d words apart contribute 1/d to the total count.

1.3.2 构造词向量和共现矩阵之间的相似关系

构建词向量和共现矩阵之间的近似关系，论文的作者提出以下的公式可以近似地表达两者之间的关系：

$$w_i^T w_j + b_i + b_j = log(X_{ij}) \tag{1}$$

其中，w_i^T **和** w_j 是**最终要求解的词向量；** b_i 和 b_j 分别是两个词向量的bias term。公式推导下文会详细介绍。构造损失函数

1.3.3 构造损失函数

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log\left(X_{ij}\right)\right)^2 \tag{2}$$

最小平方损失加了一个权重函数 $f(X_{ij})$，那么权重函数起了什么作用？我们知道在一个语料库中，肯定存在很多单词他们在一起出现的次数是很多的（frequent co-occurrences），那么我们希望：

- 1.这些单词的权重要大于那些很少在一起出现的单词（rare co-occurrences），所以这个函数要是非递减函数；
- 2.但我们也不希望这个权重过大（overweighted），当到达一定程度之后应该不再增加；
- 3.如果两个单词没有在一起出现，也就是 $X_{ij} = 0$，那么他们应该不参与到loss function的计算当中去，也就是f(x)要满足f(0)=0

满足以上两个条件的函数有很多，作者采用了如下形式的分段函数：

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

这个函数图像如下所示：

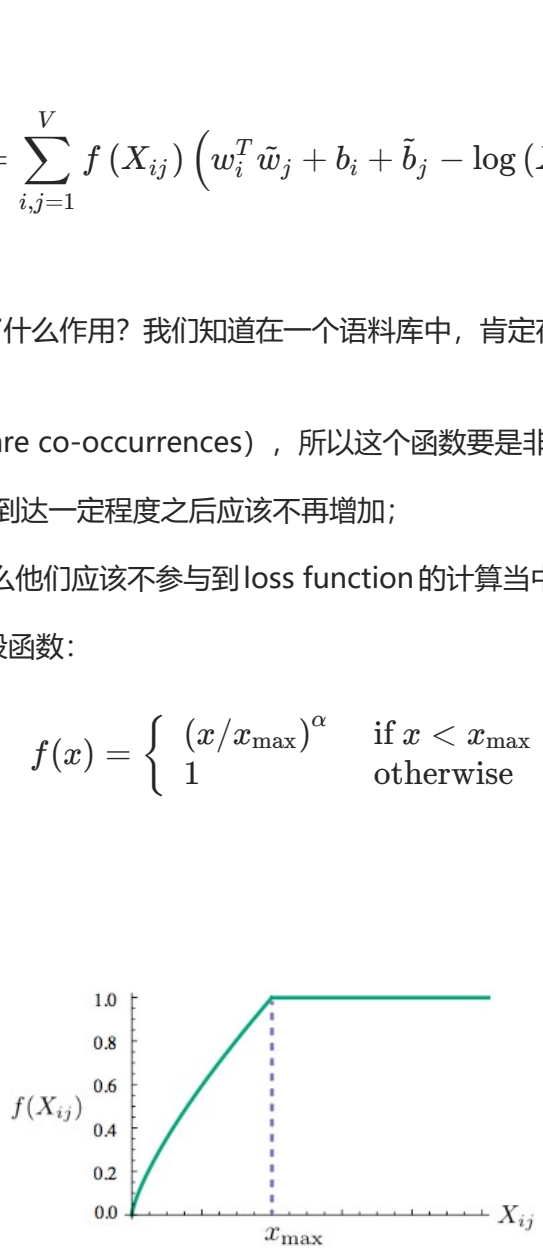


Figure 1: Weighting function f with $\alpha = 3/4$.

这篇论文中的所有实验，α的取值都是0.75，而 x_{max} 取值都是100。以上就是GloVe的实现细节，那么GloVe是如何训练的呢?

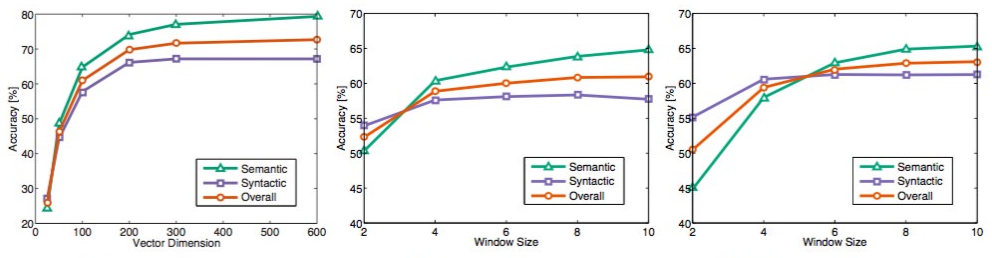
1.4 训练过程

虽然很多人声称GloVe是一种无监督（unsupervised learing）的学习方式（因为它确实不需要人工标注label），但其实它还是有label的，这个label就是公式2中的 $log(X_{ij})$，而公式2中的向量 w_i 和 w_j 就是要不断更新/学习的参数，所以本质上它的训练方式跟监督学习的训练方法没什么不一样，都是基于梯度下降的。

具体地，这篇论文里的实验是这么做的：采用了AdaGrad的梯度下降算法，对矩阵X中的所有非零元素进行随机采样，学习曲率设为0.05，在vector size小于300的情况下迭代了50次，其他大小的vectors上迭代了100次，直至收敛。

最终学习得到的是两个向量是 w_i 和 w_j，因为 X 是对称的（symmetric），所以从原理上讲 w_i 和 w_j 是也是对称的，他们唯一的区别是初始化的值不一样，而导致最终的值不一样。所以这两者其实是等价的，都可以当成最终的结果来使用。但是为了提高鲁棒性，我们最终会选择两向量之和作为最终的向量（两者的初始化不同相当于加了不同的随机噪声，所以能提高鲁棒性）。

在训练了400亿个token组成的语料后，得到的实验结果如下图所示：



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

这个图一共采用了三个指标：语义准确度，语法准确度以及总体准确度。那么我们不难发现Vector Dimension在300时能达到最佳，而context Windows size大致在6到10之间。

1.5 公式推导

我们来看一个表格：

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

理解这个表格的重点在最后一行，它表示的是两个概率的比值（ratio），我们可以使用它观察出两个单词i和j相对于单词k哪个更相关（relevant）。比如，ice和solid更相关，而stream和solid明显不相关，于是我们会发现P(solid|ice)/P(solid|steam)比1大更多。同样的gas和steam更相关，而和ice不相关，那么P(gas|ice)/P(gas|steam)就远小于1；当都有关（比如water）或者都没有关(fashion)的时候，两者的比例接近于1；这个是很直观的。

因此，以上推断可以说明**通过概率的比例而不是概率本身去学习词向量可能是一个更恰当的方法**，因此下文所有内容都围绕这一点展开。于是为了捕捉上面提到的概率比例，我们可以构造如下函数：

$$F\left(w_i, w_j, \tilde{w}_k\right)=\frac{P_{ik}}{P_{jk}} \tag{4}$$

其中，函数F的参数和具体形式未定，它有三个参数 w_i , w_j 和 $\tilde{w}_k$ ， $w \in \mathbb{R}^d$ 是词向量， $\tilde{w} \in \mathbb{R}^b$ 是上下文的词向量；因为向量空间是线性结构的，所以要表达出两个概率的比例差，最简单的办法是作差，于是我们得到：

$$F\left(w_i-w_j, \tilde{w}_k\right)=\frac{P_{ik}}{P_{jk}} \tag{5}$$

这时我们发现公式5的右侧是一个数量，而左侧则是一个向量，可以构造内积, 构造内积由两个好处，一是把向量变成标量，二是内积可以刻画相似度。左侧转换成两个向量的内积形式为：

$$F\left(\left(w_i-w_j\right)^T \tilde{w}_k\right)=\frac{P_{ik}}{P_{jk}} \tag{6}$$

我们知道共现矩阵 X 是个对称矩阵，也就是如果我们做如下交换： $w \leftrightarrow \tilde{w}_k$ ， $X \leftrightarrow X^T$ 公式6应该保持不变，显然现在的公式是不满足的。原因在于，对于左侧的式子而言，由于是向量的内积，那么能够满足 $w^T \tilde{w}=\tilde{w}^T w$ ；但是对于右侧而言，如下所示:

$$\frac{P_{ik}}{P_{jk}}=\frac{X_j X_{ik}}{X_i X_{jk}} \neq \frac{P_{ki}}{P_{kj}}=\frac{X_{ki}}{X_{kj}}$$

为了满足这个条件，首先，我们要求函数 F 要满足同态性（homomorphism，群论中的概念）：

$$F\left(\left(w_i-w_j\right)^T \tilde{w}_k\right)=\frac{F\left(w_i^T \tilde{w}_k\right)}{F\left(w_j^T \tilde{w}_k\right)} \tag{7}$$

结合公式6，我们可以得到：

$$F\left(w_i^T \tilde{w}_k\right)=P_{ik}=\frac{X_{ik}}{X_i} \tag{8}$$

接着我们发现公式7左侧是差右侧是商，我们令 $F=\exp$ ，于是我们有：

$$w_i^T \tilde{w}_k=\log\left(P_{ik}\right)=\log\left(X_{ik}\right)-\log\left(X_i\right) \tag{9}$$

此时，我们发现因为等号右侧的 $\log\left(X_i\right)$ 的存在，公式9是不满足对称性（symmetry）的，而且这个 $\log\left(X_i\right)$ 其实是跟 k 独立的，它只跟 i 有关，于是我们可以针对 w_i 增加一个偏置 b_i 把它替换掉，于是我们有：

$$w_i^T \tilde{w}_k+b_i=\log\left(X_{ik}\right) \tag{10}$$

但是公式10还是不满足对称性，于是我们针对 w_k 增加一个偏置 b_k ，从而得到公式1的形式：

$$w_i^T \tilde{w}_k+b_i+b_k=\log\left(X_{ik}\right) \tag{1}$$

1.6 总结

GloVe模型特点：

- 通过只在词与词之间的共现矩阵的非零元素上进项进行训练，使得GloVe模型有效地利于了全局统计信息
- 生成了一个有meaningful sub-structure的向量空间

Glove模型表现：

- 在给定相同语料库、词汇、窗口大小、训练时间的条件下，在词语类比任务上比word2vec表现优异
- 在结果更好的额同时速度还更快

chapter02 cs224n疑问 meaningful sub-structure是什么意思

2 词向量评估

上面讨论了诸如Word2Vec和GloVe这样的训练和发现语义空间中单词的潜在向量表示方法，下面将讨论如何评估词向量。

2.1 内部评估(Intrinsic Evaluation)

词向量的内在评估是对一组由embedding技巧(如Word2Vec和GloVe)生成的词向量在特定的中间子任务(如analog completion)，这些子任务简单而且计算速度快，因此有助于帮我们理解生成词向量的系统。内在评估应该返回一个数字，这个数字可以反映出词向量在子任务上的表现。

假设我们要创造一个使用词向量作为输入的QA系统，一个实现方式是：

1. 输入词语
2. 词语转化为词向量
3. 将词向量输入一个机器学习系统
4. 将机器学习系统输出的词向量映射成词语
5. 将词语处理成回答

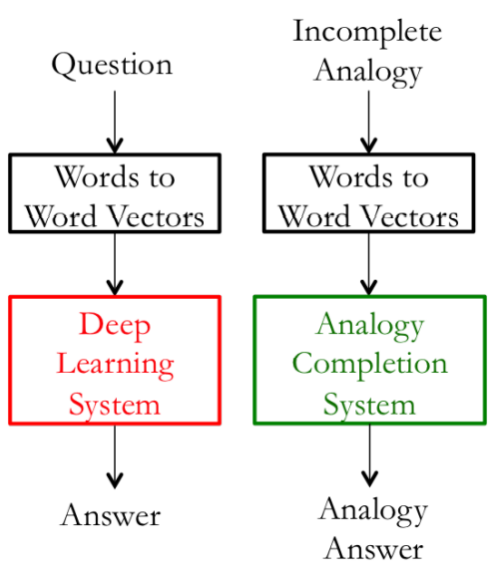


Figure 1: The left subsystem (red) being expensive to train is modified by substituting with a simpler subsystem (green) for intrinsic evaluation.

对于这样一个QA系统，显然我们需要创建最佳的词向量表示，因为它们用于下游子系统（如深层神经网络）。由于神经网络参数量过大，所以不能每次子系统(如这里的词向量)的参数就重新训练整个系统。

在这种情况下，我们想要有一个简单的能够评估词向量系统好坏程度的内在评估方法，当然，内在评估必须和最终任务表现呈正相关。

对内在评估的总结：

- 在一个具体的中间任务上评估
- 计算快
- 帮助理解子系统
- 与最终任务呈正相关

2.2 外部评估(Extrinsic Evaluation)

对外在评估的总结：

- 是真实任务的评估
- 可能计算很慢
- 判断不出出问题的部分是当前子系统、其他子系统、协同关系
- 如果替换子系统后性能变好，那么这个改变就是好的

外在系统以上的种种特点，恰恰说明了内在评估的必要性

2.3 内部评估举例：Word Vector Analogies(词向量类比评估)

2.3.1 问题定义

词向量评估的最常用的方法是word vector analogies(词向量类比)，在词向量类比中，首先有一个不完全的类比形式：

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

内在评估系统接着确定要最大化余弦相似度的词向量：

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

这个评估可以直观的解释。理想情况下，我们想要 $x_b - x_a = x_d - x_c$ (比如，queen-king=actress-actor)。该公式可推出 $x_b - x_a + x_c = x_d$ 。因此我们确定了向量 x_d ，它最大化两个词向量之间的归一化点积（即余弦相似度）。

chapter02 cs224n疑问 分母少了 $||x_i||$ ，余弦相似度的公式为 $cos\theta = \frac{\vec{a}\vec{b}}{||\vec{a}||||\vec{b}||}$

2.3.2 语义类比

2.3.2.1 答案的不唯一性

使用如词向量类比这样的评估需要注意(记住用于预训练的语料库的各个方面)，比如考虑如下形式的类比：

City1：State Containing City1 : : City2 : State Containing City2

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

Table 1: Here are semantic word vector analogies (intrinsic evaluation) that may suffer from different cities having the same name

由于美国有很多城市/城镇/乡村重名，所以州的答案不唯一。比如美国至少10个地方叫Phoenix，亚利桑那州不一定是它唯一对应的正确答案。

2.3.2.2 语料库时效性

现在考虑另外一种形式的类比：

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

Input	Result Produced
Abuja : Nigeria : : Accra	Ghana
Abuja : Nigeria : : Algiers	Algeria
Abuja : Nigeria : : Amman	Jordan
Abuja : Nigeria : : Ankara	Turkey
Abuja : Nigeria : : Antananarivo	Madagascar
Abuja : Nigeria : : Apia	Samoa
Abuja : Nigeria : : Ashgabat	Turkmenistan
Abuja : Nigeria : : Asmara	Eritrea
Abuja : Nigeria : : Astana	Kazakhstan

Table 2: Here are semantic word vector analogies (intrinsic evaluation) that may suffer from countries having different capitals at different points in time

通常情况下，答案中的city应该是最新的，比如1997年之前，哈萨克斯坦首都城市是阿拉木图。所以，如果语料库是很旧的话，还可能会出现其他各种问题。

2.3.3 句法类比

上述展示了语义层面的类比，接下来是内部评估中的句法层面的类比，分别展示了词向量捕获形容词最高级记法、过去时记法的能力。

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

Table 3: Here are syntactic word vector analogies (intrinsic evaluation) that test the notion of superlative adjectives

Input	Result Produced
dancing : danced : : decreasing	decreased
dancing : danced : : describing	described
dancing : danced : : enhancing	enhanced
dancing : danced : : falling	fell
dancing : danced : : feeding	fed
dancing : danced : : flying	flew
dancing : danced : : generating	generated
dancing : danced : : going	went
dancing : danced : : hiding	hid
dancing : danced : : hitting	hit

Table 4: Here are syntactic word vector analogies (intrinsic evaluation) that test the notion of past tense

2.4 内部评估调参举例：Analogy Evaluations(类比评估)

词向量embeding技巧(如Word2Vec和GloVe)中的超参进行探索，这些超参可以通过内部评估系统(如类比补全系统)来调整。embedding中的超参包括：

- 词向量维度
- 语料库大小
- 语料库来源/类型
- 上下文窗口大小
- 上下文对称性
- 其他

先看一下在一个类比评估任务上，同等超参下不同的embedding技巧的词向量生成方法表现如何，图2 展示了训练时间和精确度的关系，图3说明了随着语料库规模增大精确度的变化，图4说明了使用GloVe模型其他超参如何影响精确度。

一个实践技巧是，对GloVe embedding，当中心词上下文窗口大小为8时通常表现很好。

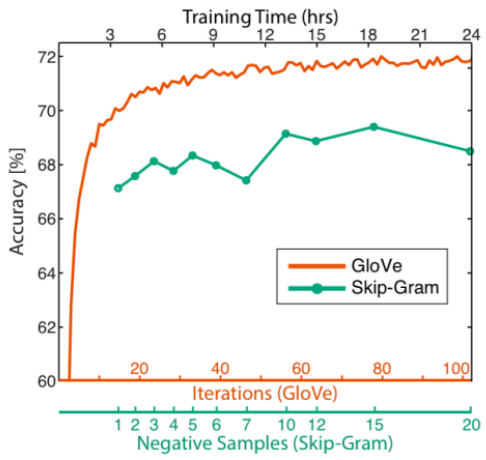


Figure 2: Here we see how training time improves training performance and helps squeeze the last few performance

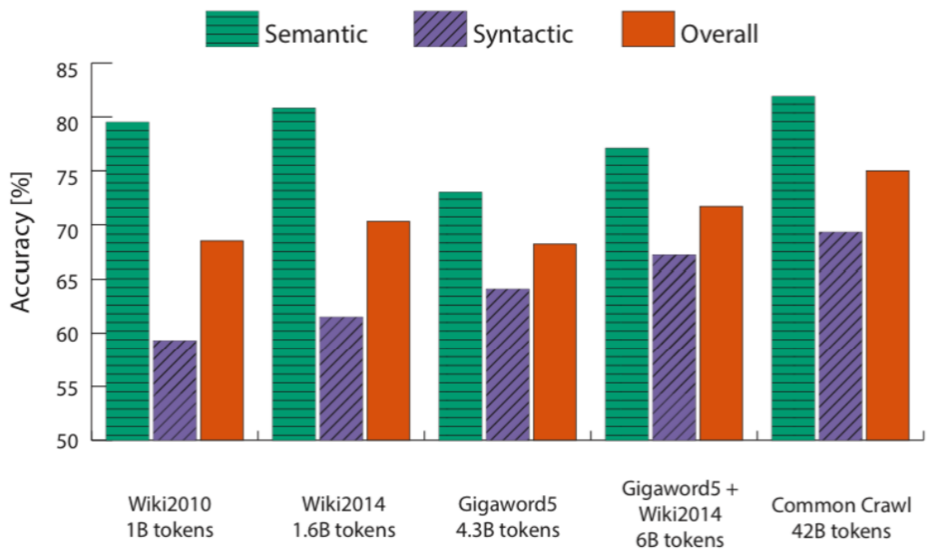
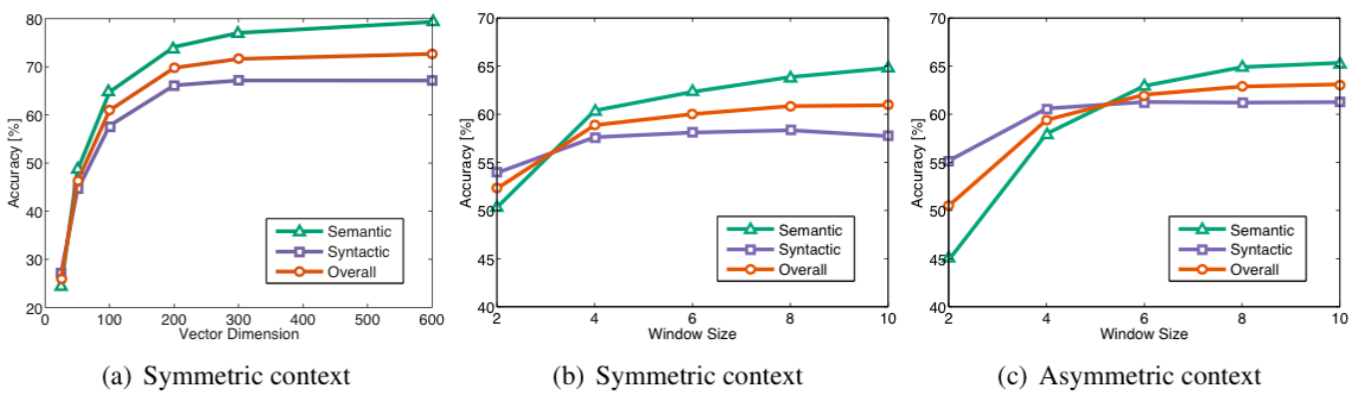


Figure 3: Here we see how performance improves with data size



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

Figure 4: We see how accuracies vary with vector dimension and context window size for GloVe

通过上述图表，我们可以有3个基本观察：

- 表现很大程度上取决于word embedding使用的模型
- 表现随着语料库增大而增大
- 当词向量维度极低时表现会变差

> 因为低维度不足以捕获单词在不同语料库中的意义，比如有4个单词："king"、"queue"、"man"、"woman", 可以直观地看到，这4个词可以由"gender"和"leardership"2个维度来编码进2-bit 的词向量，任何更低维度将难以捕获这4个词的语义信息。

2.5 内部评估举例：Correlation Evaluation(相关性评估)

另一个评估词向量质量的简单方式是， 通过让人类来判断两个词在固定取值范围(比如0-10)的相似值， 然后比较这个相似值和相应词向量的余弦相似度。这个实验已经在很多包含人类判断结果的数据集上做了，结果如table6所示。

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW	6B	57.2	65.6	68.2	57.0	32.5
SG	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	75.9	83.6	82.9	59.6	47.8
CBOW	100B	68.4	79.6	75.4	59.4	45.5

Table 6: Here we see the correlations between of word vector similarities using different embedding techniques with different human judgment datasets

2.6 进一步阅读：处理歧义

有人可能回想知道我们如何把握相同词在不同语境下的不同词向量，比如"run"这个词根据语境不同可以是名词或动词。. Improving Word Representations Via Global Context And Multiple Word Prototypes (Huang et al, 2012) 这篇论文描述了 NLP 中该如何处理这种歧义问题，核心方法是：

1. 收集所有文档中出现该词的固定大小的上下文窗口
   (比如，前面 5 个和后面 5 个)
2. 每个上下文由上下文单词向量的加权平均表示(使用 idf-weighting)
3. 由球型 k-means 来对这些上下文表示聚类
4. 最终，每个出现的单词都会重新标记为其关联的簇，并用于训练该簇的单词表示

对于这个话题的进一步详细了解，需要读原始论文。

3 词向量 SVD、Word2Vec、GloVe 对比

3.1 GloVe 与 Word2Vec 对比：

- Word2Vec 有神经网络，GloVe 没有；
- Word2Vec 关注了局部信息，GloVe 关注局部信息和全局信息；
- 都有滑动窗口但 Word2Vec 是用来训练的，GloVe 是用来统计共现矩阵的；
- GloVe 的结构比 Word2Vec 还要简单(没有神经网络)，所以速度更快；

3.2 GLoVe 与 SVD 对比

- SVD 所有单词统计权重一致，GloVe 对此进行了优化；
- GloVe 使用比值而没有直接使用共现矩阵。

当然 GloVe 看着那么好，其实并不一定，在很多任务中都没 Word2Vec 的效果好。

4 外部任务训练(Training for Extrinsic tasks)

我们目前为止聚焦了内部任务(intrinsic task)并强调了它们在开发一个好的 word embedding 上技巧的重要性，但是我们的最终目标是解决真实世界的问题，要将最终词向量用到其他的外部任务(extrinsic task)上，接下来我们将讨论处理外部任务的一般方法。

4.1 问题定义

大多数的 NLP 问题都可以定义为分类问题。比如，给一个句子，我们可以将这个句子划分为不同的不同的态度：positive、negative、neutral。相似地，在 NER(named-entity recognition，命名实体识别)中, 给定上下文和中心词，需要把中心词分到某个类别。比如 "Jim bought 300 shares of Acme Corp. in 2006", 输出结果将是：

[Jim]-Person

[Acme Corp.]Organization

[2006]Time

对于这类问题，我们通常先定义这样形式一个训练集：

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

其中 $x^{(i)}$ 是由一些word embedding技巧生成的d维向量， $y^(i)$ 是C维的one-hot向量，这个one-hot向量表明了我们希望最终预测的标签(如情感、命名实体、购买/销售决定等)

在一般的机器学习任务，我们经常把输入数据和目标的标签固定，然后使用优化技巧(如梯度下降、L-BFGS、牛顿法等)来训练权重。然而，在NLP应用中，我们引入了在训练外部任务时重新训练输入词向量的思想，下面将讨论什么时候以及为什么这样考虑。

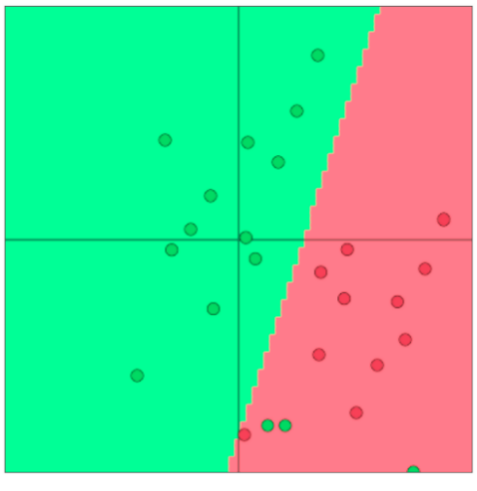


Figure 5: We can classify word vectors using simple linear decision boundaries such as the one shown here (2-D word vectors) using techniques such as logistic regression and SVMs

4.2 重新训练词向量（Retraining Word Vectors)

正如前面讨论的，我们在外部任务上使用的词向量最初是在一个简单的内部任务上优化(词向量)得到的。在很多情况下，预训练的词向量可以很好的代替在外部任务上优化词向量并且在外部任务上表现的也很好。但是，也有可能预训练的词向量需要在外部任务上进一步训练(即重新训练)以获得更好的表现，重新训练词向量是有一定风险的。

在大型训练数据集上才应该考虑重新训练重新训练词向量，对小的数据集，重新训练词向量将会导致结果变差。这是因为诸如Word2Vec或GloVe会在词空间(word space)的同一部分生成相关语义的单词, 如果在一个小的数据集上重新训练，这些单词会在词空间变换(shifted)，结果是在最终的外部任务上效果变差。

如Figure 6所示的二维空间中，词向量在外部任务上分类正确。因为训练集大小有限，我们只重新训练其中的两个向量，那么如Figure 7所示，其中一个词被误分类了因为随着词向量更新分类边界也变化了。

总结，小数据集不应该重新训练词向量，大数据集上，重新训练词向量可能会对表现有所提升。

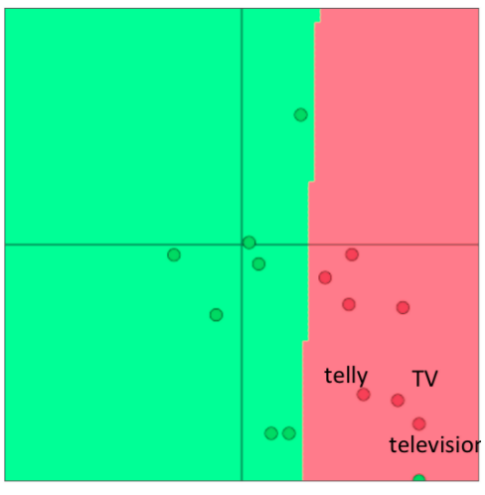


Figure 6: Here, we see that the words "Telly", "TV", and "Television" are clas□sified correctly before retraining. "Telly" and "TV" are present in the extrinsic task training set while "Television" is only present in the test set

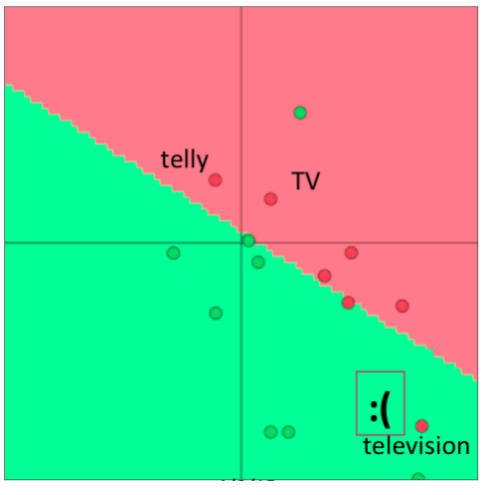


Figure 7: Here, we see that the words "Telly" and "TV" are classified correctly after training, but "Television" is not since it was not present in the training set

4.3 softmax 分类和正则化

考虑如下形式的softmax分类函数：

$$p\left(y_j=1 \mid x\right)=\frac{\exp \left(W_{j \cdot x} x\right)}{\sum_{c=1}^C \exp \left(W_c \cdot x\right)}$$

这里，我们计算词向量x在分类j中的概率。使用交叉熵损失函数，具体如下所示：

$$-\sum_{j=1}^C y_j \log \left(p\left(y_j=1 \mid x\right)\right)=-\sum_{j=1}^C y_j \log \left(\frac{\exp \left(W_{j \cdot} \cdot x\right)}{\sum_{c=1}^C \exp \left(W_c \cdot x\right)}\right)$$

上述求和项中将有C-1项是0，因为 y_j 只在一个索引位置会是1(至少现在是这样)，即x只有一个正确的分类。因此，假设k是正确分类的索引，那么可以把损失函数简化为：

$$-\log \left(\frac{\exp \left(W_k \cdot x\right)}{\sum_{c=1}^C \exp \left(W_c \cdot x\right)}\right)$$

接着将损失函数扩展到一个含N个点的数据集上：

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

唯一的不同是 $k(i)$ 是一个返回样本 $x^(i)$ 正确类别索引的函数。

下面计算更新的参数数量，计算时同时考虑模型权重 (W) 和词向量 (x) 。我们知道，一个简单的线性决策边界需要这样一个模型，该模型至少接受一个 d 维输入词向量，并生成 C 个分类上的分布。因此，要更新模型权重，我们需要更新 $C \cdot d$ 个参数。如果我们也更新词汇 V 中每个单词的词向量，那么我们将更新多达 $|V|$ 个词向量，每个都是 d 维的。因此，对于一个简单的线性分类器，参数总数将多达 $C \cdot d+|V| \cdot d$ 个：

$$\nabla_\theta J(\theta)=\begin{bmatrix} \nabla_{W_{\cdot}} \\ \vdots \\ \nabla_{W_{\cdot d}} \\ \nabla_{x_{\text{aardvark}}} \\ \vdots \\ \nabla_{x_{\text{zebra}}} \end{bmatrix}$$

考虑到模型的决策边界是多么简单，这是一个非常多的参数(如此多的参数很容易过度拟合)。

为了降低过拟合的风险，考虑加入正则项，这个正则项假设　为了减少过度拟合的风险，我们引入了一个正则化项，该项提出了贝叶斯信念(Bayesian belief)，即参数（θ）应该很小（即接近零）：

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

对有非常多参数的大模型(如神经网络模型)，正则化项更为重要。

注：正则化项是引入先验，即认为样本服从某种先验分布。

4.4 窗口分类(Window Classification)

到目前为止，我们主要探索了使用单个词向量 x 在外部任务中进行预测。实际上，由于自然语言的性质，导致这很难做到。自然语言倾向于使用同一个词来表达非常不同的含义，我们通常需要了解该词的使用上下文来区分意思。例如，如果你被要求向某人解释"to sanction" 什么意思，你会立即意识到，根据上下文，"to sanction"可能意味着“允许”或“惩罚”。在大多数情况下，我们倾向于使用一系列单词作为模型的输入。序列是一个前面和后面是上下文词向量的中心词向量。上下文中的字数也称为上下文窗口大小，上下文窗口大小要解决的问题而变化。通常，窗口越窄，句法测试的性能越好，而窗口越宽，语义测试的性能越好。

Figure 8: Here, we see a central word with a symmetric window of length 2. Such context may help disambiguate between the place Paris and the name Paris.

为了通过修改之前讨论的softmax函数模型来描述窗口分类，下面出现的公式将简单地用 $x_{window}^{(i)}$ 替换原公式中的 $x^{(i)}$ ，即：

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

梯度：

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

当然，在实现中需要梯度需要是分散的来更新相应的词向量

> The gradient will of course need to be distributed to update the corresponding word vectors in implementation

4.5 非线性分类

我们现在介绍了如神经网络之类的非线性分类器，Figure9和Figure10展示了线性分类器和非线性分类器，虽然Figure 10过拟合了，但这展示了非线性分类边界的必要性。接下来的章节将展示将神经网络作为非线性模型，这些模型在深度学习应用中表现很好。

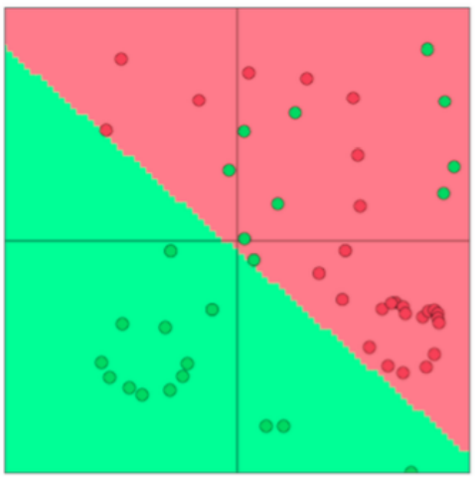


Figure 9: Here, we see that many examples are wrongly classified even though the best linear decision boundary is chosen. This is due linear decision boundaries have limited model capacity for this dataset.

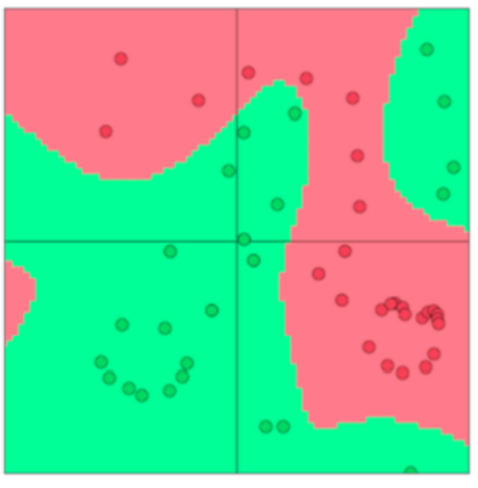


Figure 10: Here, we see that the non-linear decision boundary allows for much better classification of datapoint

5 引用

1. cs224n winter2021 notes
2. https://www.fanyeong.com/2018/02/19/glove-in-detail/
3. https://blog.csdn.net/qq_35357274/article/details/118523151
4. https://blog.csdn.net/zkq_1986/article/details/108369942