【CS224N课程笔记】词向量II: GloVe, 评估和训练

AINLP 1月27日

以下文章来源于NewBeeNLP ，作者Ryan

NewBeeNLP
永远有料，永远有趣

长按扫码关注我们

AINLP
我爱自然语言处理
一个有趣有AI的自然语言处理社区

> NewBeeNLP原创出品
> 公众号专栏作者@Ryan
> 知乎 | 机器学习课程笔记

CS224N课程笔记系列，持续更新中😋
课程主页： http://web.stanford.edu/class/cs224n/

前情提要：【CS224N课程笔记】词向量I：简介，SVD和Word2Vec

1、Global Vectors for Word Representation (GloVe)

1.1、Comparison with Previous Methods

到目前为止，我们已经研究了两类主要的词嵌入方法。第一类是基于统计并且依赖矩阵分解（例如，LSA，HAL）。虽然这类方法有效地利用了全局的信息，它们主要用于捕获单词的相似性，但是对类似单词类比的任务上表现不好。另外一类方法是基于浅层窗口（例如，$Skip-Gram$和 $CBOW$ 模型），这类模型通过在局部上下文窗口通过预测来

学习词向量。这些模型除了在单词相似性任务上表现良好外，还展示了捕获复杂语言模式能力，但未能利用到全局共现统计数据。

相比之下，$GloVe$ 由一个加权最小二乘模型组成，在全局词-词共现统计上训练，这样能有效地利用全局统计数据。这个模型生成了有意义的子结构的单词向量空间，在词类比任务上表现非常出色。$GloVe$ 利用全局统计量，以最小二乘为目标，预测单词 j 出现在单词 i 上下文中的概率。

1.2、Co-occurrence Matrix

令 X 表示为词-词共现矩阵，其中 X_{ij} 表示词 j 出现在词 i 的上下文的次数。令 $X_i = \sum X_{ij}$ 为任意词 k 出现在词 i 的上下文的次数。最后，令$P_{ij} = P(w_j|w_i) = \frac{X_{ij}}{X_i}$ 是词j出现在词 i 的上下文的概率。计算这个矩阵需要遍历一次整个语料库获得统计信息。对庞大的语料库，这样的遍历会产生非常大的计算量，但是这只是一次性的前期投入成本。

1.3、Least Squares Objective

回想一下 $Skip-Gram$ 模型，我们使用 $softmax$ 来计算词 j 出现在词 i 的上下文的概率。

$$Q_{ij} = \frac{exp(u_j^T v_i)}{\sum_{w=1}^{W} exp(u_w^T v_i)}$$

训练时以在线随机的方式进行，但是暗含全局交叉熵损失可以如下计算：

$$J = -\sum_{i \epsilon corpus} \sum_{j \epsilon context(i)} logQ_{ij}$$

同样的单词 i 和 j 可能在语料库中出现多次，因此首先将 i 和 j 相同的值组合起来更有效：

$$J = -\sum_{i=1}^{W} \sum_{j=1}^{W} X_{ij} logQ_{ij}$$

其中共现频率的值是通过共现矩阵 X 给定。交叉熵损失的一个显着缺点是要求分布 Q 被正确归一化，因为对整个词汇的求和的计算量是非常大的。因此，我们使用一个最小二乘的目标函数，其中 P 和 Q 的归一化因子被丢弃了：

$$\hat{J} = \sum_{i=1}^{W} \sum_{j=1}^{W} X_i (\hat{P}_{ij} - \hat{Q}_{ij})^2$$

其中 $\widehat{P}_{ij} = X_{ij}$ 和 $\widehat{Q}_{ij} = exp(u_j^T v_i)$ 是非归一化分布。这个公式带来了一新的问题，X_{ij} 经常会是很大的值，从而难以优化。一个有效的改变是最小化 $\widehat{P}$ 和 $\widehat{Q}$ 对数的平方误差：

$$\begin{aligned}\widehat{J} &= \sum_{i=1}^{W}\sum_{j=1}^{W} X_i(log(\widehat{P}_{ij}) - log(\widehat{Q}_{ij}))^2 \\ &= \sum_{i=1}^{W}\sum_{j=1}^{W} X_i(u_j^T v_i - logX_{ij})^2\end{aligned}$$

另外一个问题是权值因子 X_i 不能保证是最优的。因此，我们引入更一般化的权值函数，我们可以自由地依赖于上下文单词：

$$\widehat{J} = \sum_{i=1}^{W}\sum_{j=1}^{W} f(X_{ij})(u_j^T v_i - logX_{ij})^2$$

1.4、Conclusion

总而言之，$GloVe$ 模型仅对单词共现矩阵中的非零元素训练，从而有效地利用全局统计信息，并生成有意义子结构的向量空间。给出相同的语料库，词汇，窗口大小和训练时间，它的表现都优于 $Word2Vec$，它可以更快地实现更好的效果，并且无论速度如何，都能获得最佳效果。

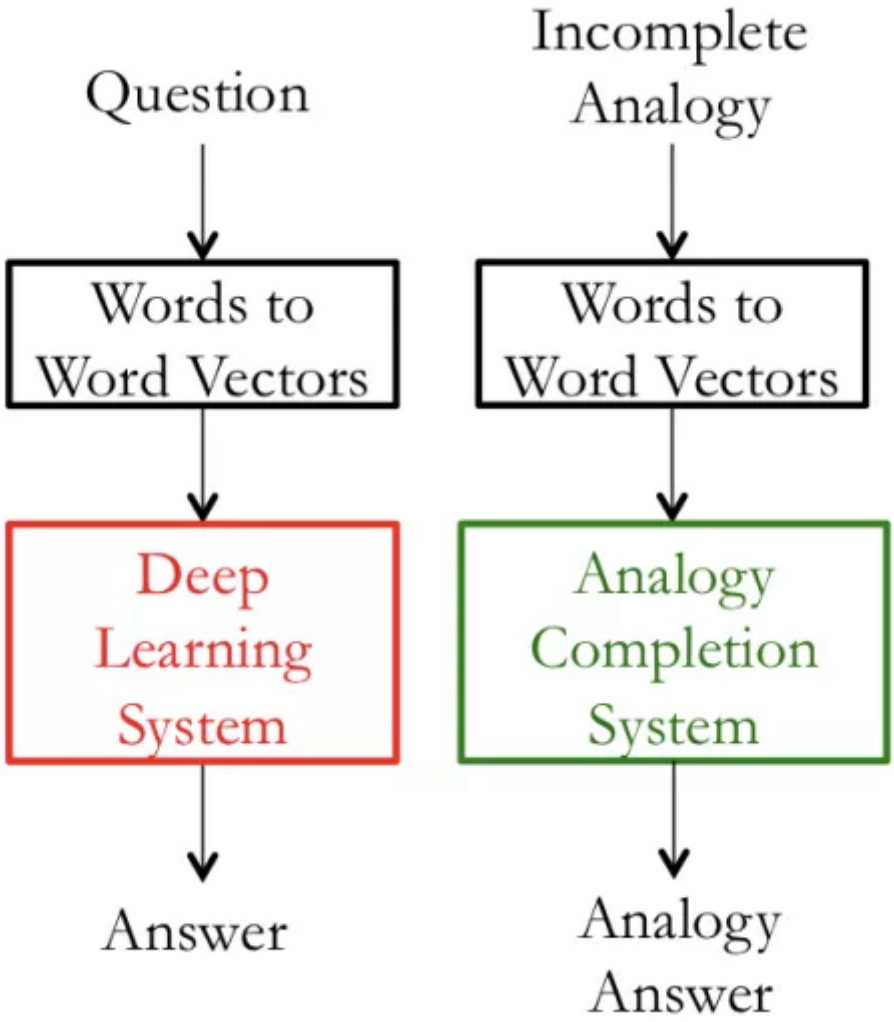
2、Evaluation of Word Vectors

到目前为止，我们已经讨论了诸如 $Word2Vec$ 和 $GloVe$ 来训练和发现语义空间中的自然语言词语的潜在向量表示。在这部分，我们讨论如何量化评估词向量的质量。

2.1、Intrinsic Evaluation

词向量的内部评估是对一组由如 $Word2Vec$ 或 $GloVe$ 生成的词向量在特定的中间子任务（如词类比）上的评估。这些子任务通常简单而且计算速度快，从而能够帮助我们理解生成的的词向量。内部评估通常应该返回给我们一个数值，来表示这些词向量在评估子任务上的表现。

在下图中，左子系统（红色）训练的计算量大，因此更改为一个简单的子系统（绿色）作内部评估。



「动机：」 我们考虑一个目标是创建一个问答系统，其中使用词向量作为输入的例子。一个方法是训练一个机器学习系统：

1. 输入词
2. 将输入词语转换为词向量
3. 对一个复杂的机器学习系统，使用词向量作为输入
4. 将输出的词向量通过系统映射到自然语言词语上。
5. 生成词语作为答案

当然，在训练这样的一个问答系统的过程中，我们需要创建最有的词向量表示，因为它们被用在下游子系统（例如深度神经网络）。在实际操作中，我们需要对 $Word2Vec$ 子系统中的许多超参数进行调整（例如词向量的维度）。虽然最理想的方法是在 $Word2Vec$ 子系统中的任何参数改变后都重新训练训练，但从工程角度来看是不实际的，因为机器学习系统（在第3步）通常是一个深层神经网络，网络中的数百万个参数需要很长的时间训练。在这样的情况下，我们希望能有一个简单的内部评估技术来度量词向量子系统的好坏。显然，一个要求是内部评价与最终任务的表现有正相关关系。

2.2、Extrinsic Evaluation

词向量的外部评估是对一组在实际任务中生成的词向量的评估。这些任务通常复杂而且计算速度慢。对我们上面的例子，允许对问题答案进行评估的系统是外部评估系统。通常，优化表现不佳的外部评估系统我们难以确定哪个特定子系统存在错误，这就需要进一步的内部评估。

2.3、Intrinsic Evaluation Example: Word Vector Analogies

一个比较常用的内部评估的方法是词向量的类比。在词向量类比中，给定以下形式的不完整类比：

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

然后内部评估系统计算词向量的最大余弦相似度：

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{|x_b - x_a + x_c|}$$

这个指标有直观的解释，理想情况下，我们希望 $x_b - x_a = x_d - x_c$ （例如 $queen - king = actress - actor$）。这就暗含这我们希望 $x_b - x_a + x_c = x_d$ 。因此，我们确定可以最大化两个词向量之间的归一化点积的向量 x_d 即可（即余弦相似度）。使用诸如词向量类比的内部评估技术应该小心处理（要考虑到预训练的语料库的各个方面）。例如，考虑以下的类比形式：

$$City\,1 : State\,containing\,City\,1 :: City\,2 : State\,containing\,City\,2$$

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

上图是可能受到具有相同名称的不同城市的语义词向量类比（内在评估）。在上面很多的例子，美国有很多同名的城市 / 城镇 / 村庄。因此，很多州都符合正确的答案。例如，在

美国至少有 10 个地方的名称是 $Phoenix$，所以 $Arizona$ 不是唯一的正确答案。在考虑以下的类比形式：

$$Capital\,City\,1 : Country\,1 :: Capital\,City\,2 : Country\,2$$

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

上图是可能在不同时间点有不同首都的国家的语义词向量类比（内在评估）。上面很多的例子，这个任务中生成的城市仅仅是近期的国家首都。例如，1997 年之前 $Kazakhstan$ 的首都是 $Almaty$。因此，如果我们的语料库过时就会出现问题。

之前的两个例子说明如何使用词向量进行语义测试。我们也可以使用词向量类似进行语法测试。以下内部评估测试词向量来捕获形容词的最高级概念的能力，如下图所示：

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

类似地，下图的内部评估展示了测试词向量捕获过去时态概念的能力：

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

2.4、Intrinsic Evaluation Tuning Example: Analogy Evaluations

我们现在探讨使用内在评估系统（如类比系统）来调整的词向量嵌入技术（如 $Word2Vec$ 和 $GloVe$）中的超参数。我们首先来看看在类比评估任务中，在相同的超参数下，由不同方法创建的词向量表现效果：

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

根据上表，我们可以看到 3 点：

- **「模型的表现高度依赖使用词向量的模型：」**

这点是可以预期到的，因为不同的生成词向量方法是基于不同的特性的（例如共现技术，奇异向量等等）。

- **「语料库更大模型的表现更好：」**

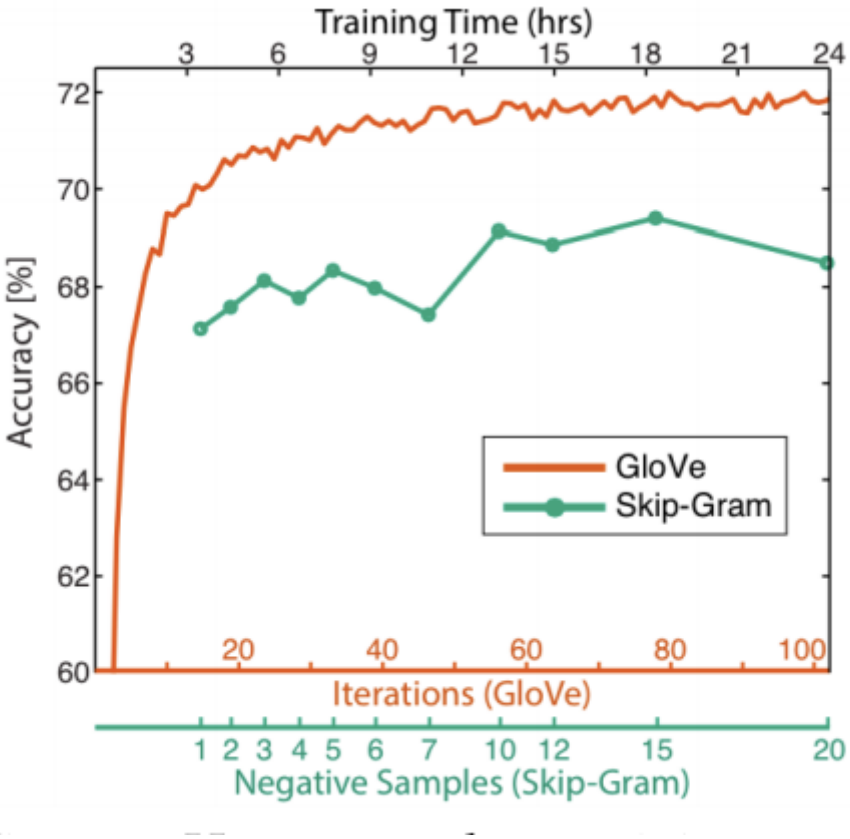
这是因为模型训练的语料越大，模型的表现就会更好。例如，如果训练的时候没有包含测试的词语那么词类比会产生错误的结果。

- **「对于极高或者极低维度的词向量，模型的表现较差：」**

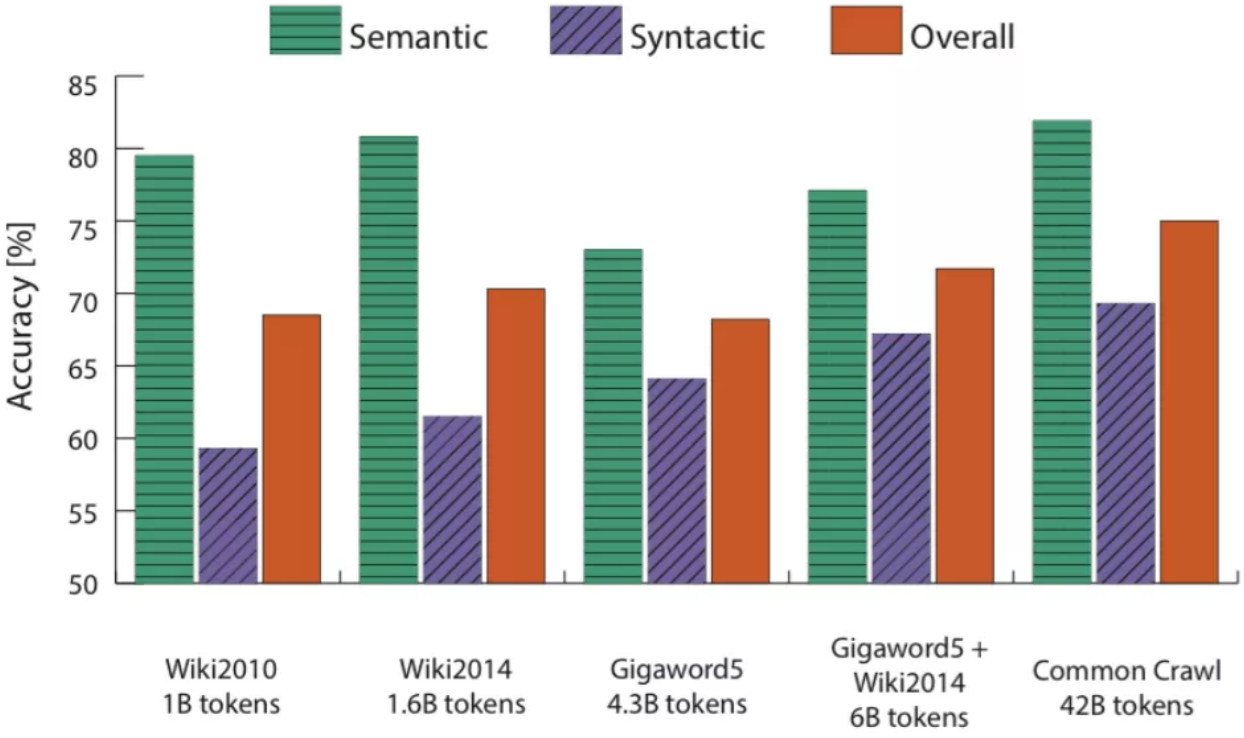
低维度的词向量不能捕获在语料库中不同词语的意义。这可以被看作是我们的模型复杂度太低的高偏差问题。例如，我们考虑单词 “$king$”、“$queen$”、“man”、“$woman$”。直观上，我们需要使用例如 “性别” 和 “领导” 两个维度来将它们编码成 2 字节的词向量。维度较低的词向量不会捕获四个单词之间的语义差异，而过高的维度的可能捕获语料库中无助于泛化的噪声-即所谓的高方差问题。

「tips:」 $GloVe$ 中使用中心词左右的窗口大小为 8 时模型表现较好。

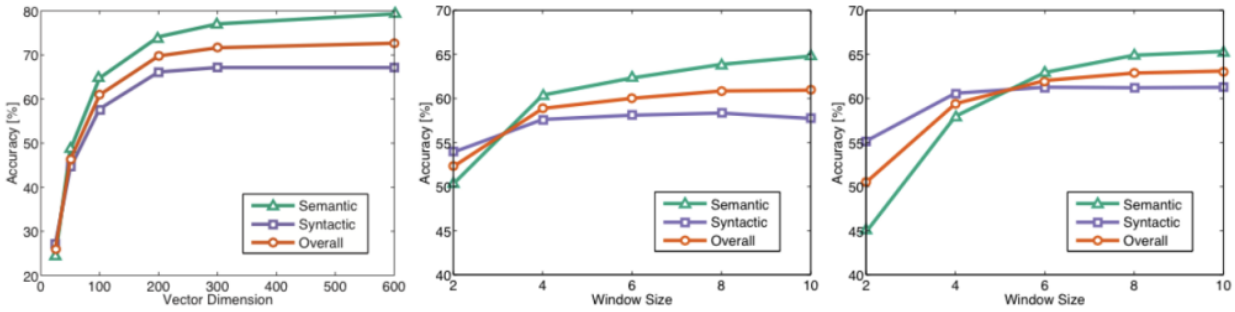
下图可以看到训练时间对模型表现的影响



下图可以看到增加语料库规模对模型准确度的影响



下图可以看到不同超参数对 $GloVe$ 模型准确度的影响



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

2.5、Intrinsic Evaluation Example: Correlation Evaluation

另外一个评估词向量质量的简单方法是，让人去给两个词的相似度在一个固定的范围内（例如 $0-10$）评分，然后将其与对应词向量的余弦相似度进行对比。这已经在包含人为评估的各种数据集上尝试过。

下图是不同方法生成的词向量和认为评估的相关性：

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

2.6 Further Reading: Dealing With Ambiguity

我们想知道如何处理在不同的自然语言处理使用场景下，用不同的的词向量来捕获同一个单词在不同场景下的不同用法。例如，“run”是一个名词也是一个动词，在不同的语境

中它的词性也会不同。论文《Improving Word Representations Via Global Context And Multiple Word Prototypes》提出上述问题的的解决方法。该方法的本质如下：

1. 对所有出现的词，收集其固定大小上下文窗口（例如，前 5 个和后 5 个）。
2. 每个上下文使用上下文词向量的加权平均值来表示。
3. 用球面 $k-means$ 对这些上下文表示进行聚类。
4. 最后，每个出现的单词都重新标签为其相关联的类，同时对这个类，来训练对应的词向量。

3、Training for Extrinsic Tasks

到目前为止，我们一直都关注于内在任务，并强调其在开发良好的词向量技术中的重要性。但是大多数实际问题的最终目标是将词向量结果用于其他的外部任务。接下来会讨论处理外部任务的方法。

3.1、Problem Formulation

很多 NLP 的外部任务都可以表述为分类任务。例如，给定一个句子，我们可以对这个句子做情感分类，判断其情感类别为正面，负面还是中性。相似地，在命名实体识别（NER），给定一个上下文和一个中心词，我们想将中心词分类为许多类别之一。对输入，“Jim bought 300 shares of Acme Corp. in 2006”，我们希望有这样的一个分类结果：“$[Jim]_{Person}\ bought\ 300\ share\ of\ [Acme\ Corp.]_{Organization}\ in\ [2006]_{Time}$”。

对这类问题，我们一般有以下形式的训练集：$\{x^{(i)}, y^{(i)}\}_1^N$，其中 $x^{(i)}$ 是一个 d 维的词向量，$y^{(i)}$ 是一个 C 维的 $one-hot$ 向量，表示我们希望最终预测的标签（情感，其他词，专有名词，买 / 卖决策等）。

我们可以使用诸如逻辑回归和 SVM 之类的算法对 $2-D$ 词向量来进行分类，如下图所示

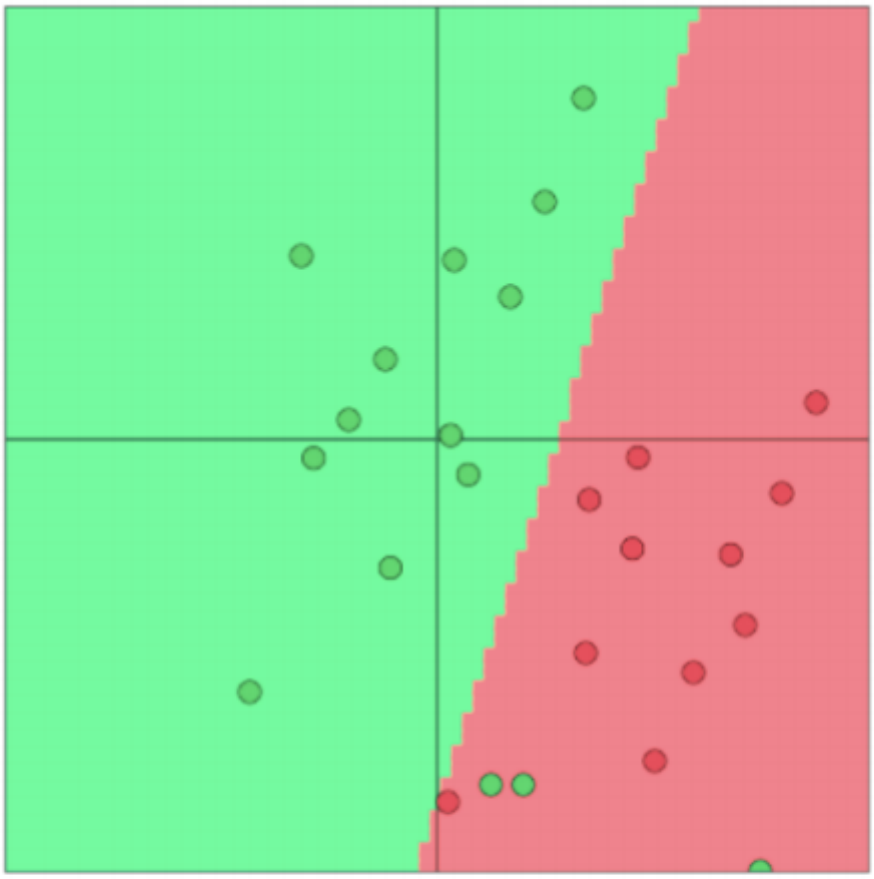


Figure 5: We can classify word vectors using simple linear decision boundaries such as the one shown here (2-D word vectors) using techniques such as logistic regression and SVMs

在一般的机器学习任务中，我们通常固定输入数据和目标标签，然后使用优化算法来训练权重（例如梯度下降，$L-BFGS$，牛顿法等等）。然而在 NLP 应用中，我们引入一个新的想，当训练外部任务的时候，重新训练输入的词向量。下面我们讨论何时使用和为什么要这样做。

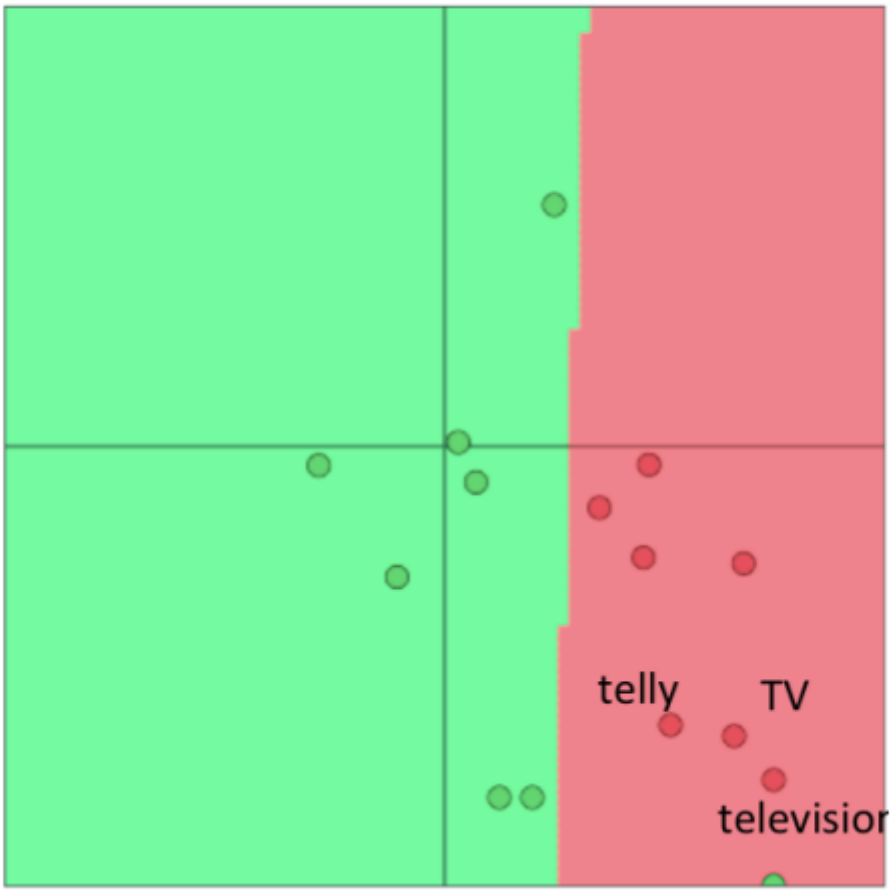
3.2、Retraining Word Vectors

正如我们迄今所讨论的那样，我们用于外部评估的词向量是通过一个简单的内部评估来进行优化并初始化。在许多情况下，这些预训练的词向量在外部评估中表现良好。但是，这些预训练的词向量在外部评估中的表现仍然有提高的可能。然而，重新训练存在着一定的风险。

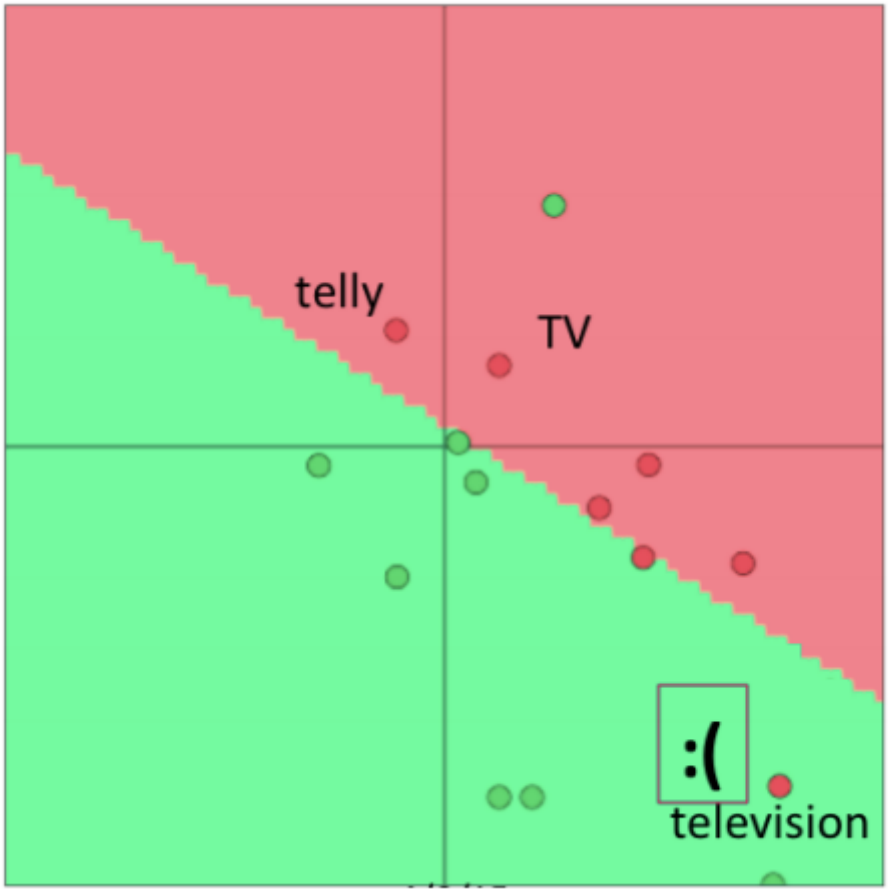
如果我们在外部评估中重新训练词向量，这就需要保证训练集足够大并能覆盖词汇表中大部分的单词。这是因为 $Word2Vec$ 或 $GloVe$ 产生语义相关的词以位于单词空间的相同

部分。如果我们在很小的词汇表上重新训练这些词，这些词就会在词空间移动，因此在最终的任务上，模型的 $performance$ 反而会降低。

让我们使用一个例子来探讨上述的问题，下图中单词 “$Telly$”，“TV”和“$Television$”在重新训练前的分类是正确的。



现在因为训练集的大小受到限制，“$Telly$”，“TV”出现在外部评估的训练集中，但是“$Television$”只在测试集中出现。因此在下图中，可以看到重新训练后“$Telly$”，“TV”分类正确，但是“$Television$”分类错误，原因在于“$Television$”没有出现在训练集中。



因此，如果在训练集很小的情况下，词向量不应该重新训练。如果训练集很大，重新训练可以提升最终表现。

3.3、Softmax Classification and Regularization

我们考虑使用 $Softmax$ 分类函数，函数形式如下所示：

$$(y_j = 1 \mid x) = \frac{exp(W_{j.}x)}{\sum_{c=1}^{C} exp(W_{c.}x)}$$

这里我们计算词向量 x 是类别 j 的概率。使用交叉熵损失函数，计算一个样例的损失如下所示：

$$-\sum_{j=1}^{C} y_j log(p(y_j = 1 \mid x)) = -\sum_{j=1}^{C} y_i log(\frac{exp(W_{j.}x)}{\sum_{c=1}^{C} exp(W_{c.}x)})$$

当然，上述求和是对 $(C-1)$ 个零值求和，因为 y_i仅在单个索引为 1，这意味着 x 仅属于 1 个正确的类别。现在我们定义 k 为正确类别的索引。因此，我们现在可以简化损失函数：

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

然后我们可以扩展为有 N 个单词的损失函数：

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

上面公式的唯一不同是 $k(i)$ 现在一个函数，返回 $x^{(i)}$ 对应的每正确的类的索引。

现在我们来估计一下同时训练模型的权值 (W) 和词向量 (x) 时参数的树木。我们知道一个简单的线性决策模型至少需要一个 d 维的词向量输入和生成一个 C 个类别的分布。因此更新模型的权值，我们需要 $C \cdot d$ 个参数。如果我们也对词汇表 V 中的每个单词都更新词向量，那么就要更新 $|V|$ 个词向量，每一个的维度是 d 维。因此对一个简单的线性分类模型，总共的参数数目是 $C \cdot d + |V|$ ：

$$\nabla_{\theta}J(\theta) = \begin{bmatrix} \nabla_{W_{.1}} \\ \vdots \\ \nabla_{W_{.d}} \\ \nabla_{aardvark} \\ \vdots \\ \nabla_{zebra} \end{bmatrix}$$

对于一个简单的模型来说，这是相当大的参数量-这样的参数量很可能会出现过拟合的问题。

为了降低过拟合的风险，我们引入一个正则项，从贝叶斯派的思想看，这个正则项是对模型的参数加上一个先验分布，让参数变小（即接近于 0）：

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

如果调整好目标权重 λ 的值，最小化上面的函数将会降低出现很大的参数值的可能性，同时也提高模型的泛化能力。在我们使用更多参数更复杂的模型（例如神经网络）时，就更加需要正则化的思想。

3.4、Window Classification

下图是我们有一个中心词和一个长度为 2 的对称窗口。这样的上下文可以帮助分辨 $Paris$ 是一个地点还是一个名字。

目前为止，我们主要探讨了使用单个单词向量 x 预测的外部评估任务。在现实中，因为自然语言处理的性质，这几乎不会有这样的任务。在自然语言处理中，常常存在着一词多义的情况，我们一般要利用词的上下文来判断其不同的意义。例如，如果你要某人解释“$to\ sanction$”是什么意思，你会马上意识到根据“$to\ sanction$”的上下文其意思可能是“$to\ permit$”或者“$to\ punish$”。在更多的情况下，我们使用一个词序列作为模型的输入。这个序列是由中心词向量和上下文词向量组成。上下文中的词的数量也被称作是上下文窗口大小，其大小在不同的任务中取不同的值。一般来说，窄窗口在句义测试中表现较好，但是在语义测试中宽窗口的表现较好。

为了将之前讨论的 $Softmax$ 模型修改为使用词窗口来进行分类，我们只需要按照下面形式将 $x^{(i)}$ 替换为 $x^{(i)}_{window}$：

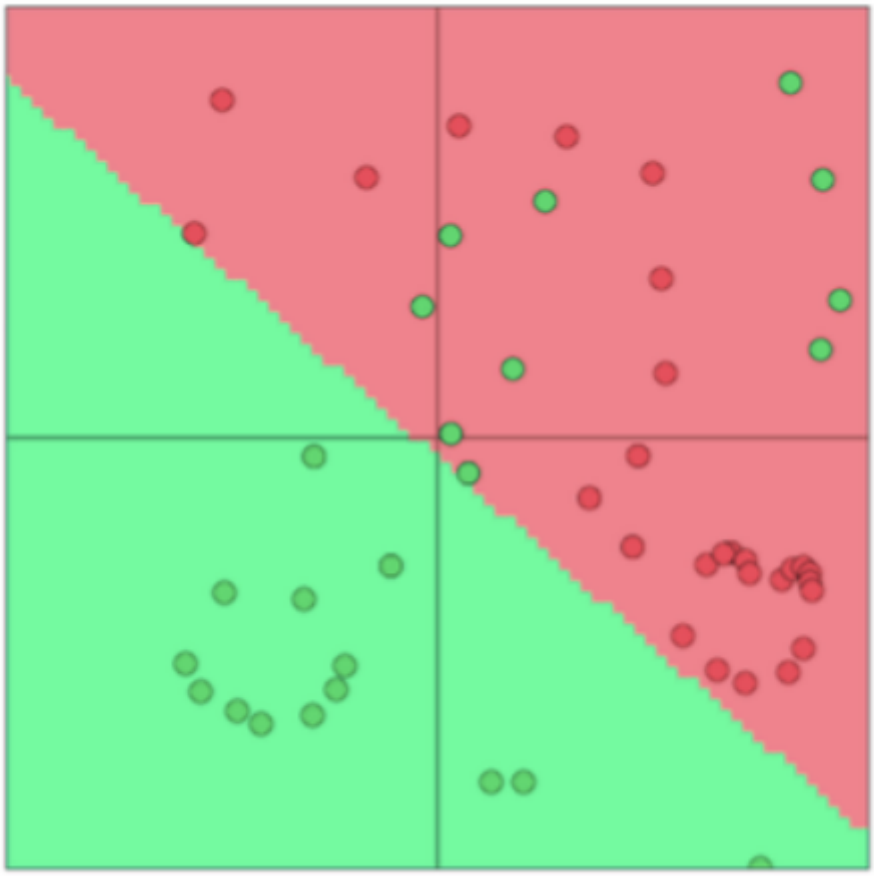
$$x^{(i)}_{window} = \begin{bmatrix} x^{(i-2)} \\ x^{(i-1)} \\ x^{(i)} \\ x^{(i+1)} \\ x^{(i+2)} \end{bmatrix}$$

因此，当我们计算单词的损失梯度如下所示，当然需要分配梯度来更新相应的词向量：

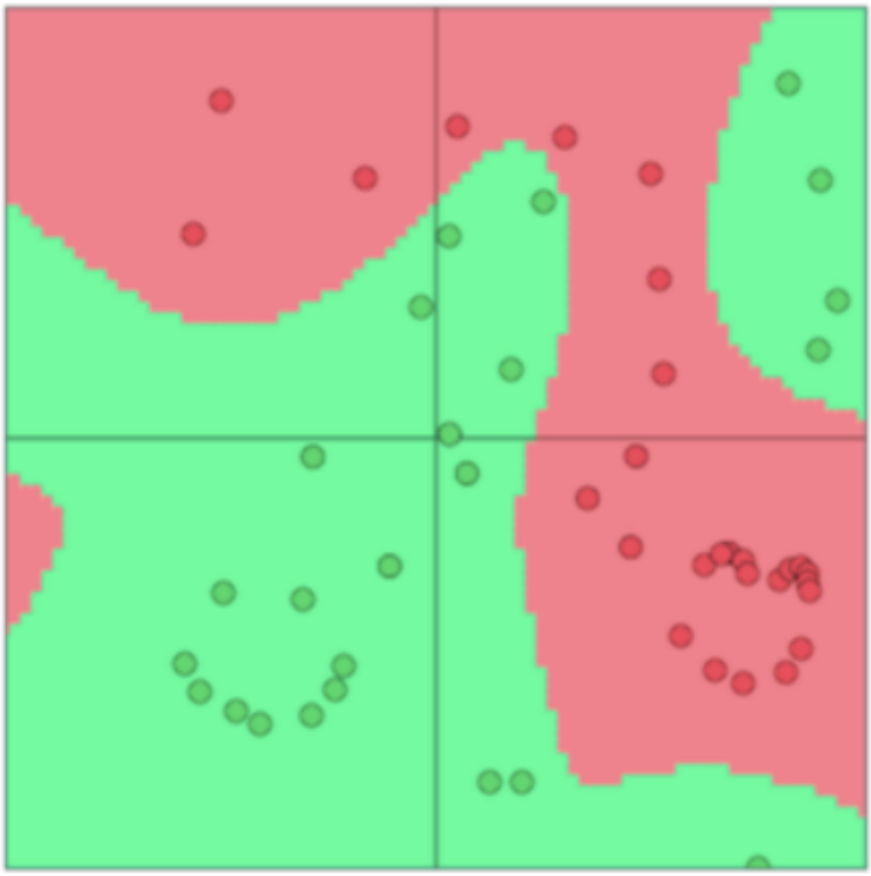
$$x^{(i)}_{window} = \begin{bmatrix} \nabla_{x^{(i-2)}} \\ \nabla_{x^{(i-1)}} \\ \nabla_{x^{(i)}} \\ \nabla_{x^{(i+1)}} \\ \nabla_{x^{(i+2)}} \end{bmatrix}$$

3.5、Non-linear Classifiers

我们现在介绍一下非线性分类模型，例如神经网络。在下图中，我们看到即使是最优的线性分类平面，也有许多样例都被错误的分类。这是因为线性模型在这个数据集上的分类能力有限。



在下图中，我们看到非线性分类模型可以对上面的数据集的样例有着更好的分类结果，这个简答的例子可以初步的说明我们为什么需要非线性模型。



- END -

由于微信平台算法改版，公号内容将不再以时间排序展示，如果大家想第一时间看到我们的推送，强烈建议星标我们和给我们多点点【在看】。星标具体步骤为：

（1）点击页面**最上方"AINLP"**，进入公众号主页。
（2）点击**右上角的小点点**，在弹出页面点击“**设为星标**”，就可以啦。
感谢支持，比心❤️。

欢迎加入**AINLP**技术交流群