论文｜Sentence2Vec & GloVe 算法原理、推导与实现

原创 Thinkgamer 搜索与推荐Wiki 2020-12-21

收录于话题
#论文笔记

18个

▼ 相关推荐 ▼

1、3W字长文带你轻松入门视觉Transformer
2、近100页的《常见的五种神经网络》汇总电子书
3、美团点评｜深度学习在推荐中的实践
4、聊一聊海量公众号下我是如何进行筛选和内容消费的

万物皆可Embedding系列会结合论文和实践经验进行介绍，前期主要集中在论文中，后期会加入实践经验和案例，目前已更新：

- 万物皆可Vector之语言模型：从N-Gram到NNLM、RNNLM
- 万物皆可Vector之Word2vec：2个模型、2个优化及实战使用
- Item2vec中值得细细品味的8个经典tricks和thinks
- Doc2vec的算法原理、代码实现及应用启发
- Sentence2Vec & GloVe 算法原理、推导与实现

后续会持续更新Embedding相关的文章，欢迎持续关注「搜索与推荐Wiki」，戳文末「**阅读原文**」触达更多有关「推荐系统」笔记！

Sentence2vec

Sentence2vec 是2017年发表于ICLR（国际学习展示回忆）的一篇论文，其全称为：**A Simple but tough-to-beat baseline for sentence embeddings**

下面来看一下论文所介绍的内容（论文的内容比较晦涩难懂，小编水平也不高，如果不当之处，评论区留言，感谢！）。

1、概述

论文主要提出了一种无监督的，基于单词词向量计算句子embedding的方法，称之为Smooth Inverse Frequecy(SIF)，使用加权平均的方法从word embedding到sentence embedding，然后再基于句子的embedding进行相似度计算，下面使用Sentence2vec来代替模型的思想。

论文中提出的计算方法比直接平均求句子embedding的方法效果要好一些，在一些任务中甚至比RNN、LSTM模型表现还要好一些。

与该论文中思路比较相近的做法有：

- 直接使用词语的平均embedding来表示句子，即不带权平均
- 使用TF-IDF值作为权重，进行带权计算（前提是句子中没有大量重复的词语，且使用tfidf值作为权重没有理论依据）

通过引入SIF，结合词语embedding加权计算句子embedding，不仅表现较好，而且有较好的鲁棒性，论文中提到了三点：

- 使用不同领域的语料训练得到的不同词语embedding，均取得了不错的效果，说明算法对各种语料都比较友好
- 使用不同语料计算得到的词频，作为词语的权重，对最终的结果影响很小
- 对于方法中的超参数，在很大范围内，获得的结果都是区域一致的，即超参数的选择没有太大的影响

2、理论

a）潜在变量生成模型

在介绍Sentence2vec之前，先看一下潜在变量生成模型（latent variable generative model），其将语料的生成过程看作是一个动态的过程，第t个单词是在第t步生成的，每个单词w对应一个实值向量R^d。这个动态过程是通过discourse vector $c_t \in R^d$随机游走驱动的。discourse vector表达的是what is being talked about。

discourse vector c_t和单词的向量v_w的内积表达的是discourse和word之间的相关性，并且假设t时间观测到的w的概率为这个内积的对数线性关系（log linear），表达式为：

$$Pr[w\,emitted\,at\,time\,t|c_t] \propto exp(< c_t, v_w >)$$

由于 c_t 是较小幅度的随机游走生成的， c_t 和 c_{t+1} 之间只是相差一个很小的随机差向量，因此相邻的单词由相似的discourses vector 生成，另外计算表明这种模型的随机游走允许偶尔 c_t 有较大的 jump，通过这种方法生成的词向量，与word2vec(CBOW)和Glove是相似的。

b）Sentence2vec 在随机游走上的改进

在给定句子s的情况下，对控制该句子的向量 discourse vector 进行最大似然估计，我们观察到在句子生成单词的时候，discourse vector c_t 变化特别小，为了简单起见，认为其是固定不变的，为 c_s，可以证明 对 c_s 的最大似然估计就是该句中所有单词向量的平均。

Sentence2vec对模型的改进为增加了两项平滑（smoothing term），原因是：有些单词在上下文之外出现，可能会对discourse vector产生影响；有些常见的停用词和discourse vector几乎没有关系。

两项平滑技术为：

- 1、在对数线性模型中引入 了累加项 $\alpha p(w)$，$p(w)$ 表示的是单词w在整个语料中出现的概率，α 是一个超参数，这样即使和c_s的向量内积很小，这个单词也有概率出现
- 2、引入纠错项 $c_0 \in R^d$ (a common discourse vector)，其意义是句子的最频繁的意义可以认为是句子的最重要组成部分。常常可以与语法联系起来. 文章中认为对于某个单词，其沿着c_0方向的成分较大(即向量投影更长)，这个纠正项就会提升这个单词出现的概率.

纠正后的单词w在句子s中出现的概率为：

$$Pr[w\,emitted in sentences|c_s] = \alpha p(w) + (1-\alpha)\frac{exp(< \tilde{c_s}, v_w >)}{Z_{\tilde{c_s}}}$$

其中：

- $\tilde{c_s} = \beta c_0 + (1-\beta)c_s, c_0 \perp c_s$
- α, β为超参数
- $Z_{\tilde{c_s}} = \sum_{w \in V} exp(< \tilde{c_s}, v_w >)$ 是归一化常数

从上面的公式中也可以看出，一个与c_s 没有关系的词语w 也可以在句子中出现，因为：

- $\alpha p(w)$ 常数项
- 与 common discourse vector c_0的相关性

c）计算句子相关性

句子的向量，即上文提到的c_s 可以通过最大似然函数去生成，这里假设组成句子的词语 v_w是统一分散的，因此这里归一化Z_c 对于不同句子的值都是大致相同的，即对于任意的$\tilde{c_s}$，Z值都是相同的，，在这个前提下，得到的似然函数为：

$$p[s|c_s] = \prod_{w \in s} p(w|c_s) = \prod_{w \in s} [\alpha p(w) + (1-\alpha)\frac{exp(< v_w, \tilde{c_s} >)}{Z}]$$

取对数，可得：

$$f_w(\tilde{c_s}) = log[\alpha p(w) + (1-\alpha)\frac{exp(< v_w, \tilde{c_s} >)}{Z}]$$

经过一系列推导，可得最终的目标函数为：

$$argmax \sum_{w \in s} f_w(\tilde{c_s})$$

其正比于：

$$\sum_{w \in s} \frac{\alpha}{p(w)+\alpha} v_w$$

其中$\alpha = \frac{1-\alpha}{\alpha Z}$

因此可以得到:

- 最优解为句子中所有单词向量的加权平均
- 对于词频更高的单词w, 权值更小，因此这种方法也等同于下采样频繁单词

最后，为了得到最终的句子向量c_s，我们需要估计c_0.。通过计算向量$\tilde{c_s}$的first principal component(PCA中的主成分)，将其作为c_0，最终的句子向量即为$\tilde{c_s}$减去主成份向量c_0。

d）整体算法流程

Algorithm 1 Sentence Embedding

Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter a and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.
Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$

1: **for all** sentence s in $\mathcal{S}$ **do**
2: $\quad v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$
3: **end for**
4: Form a matrix X whose columns are $\{v_s : s \in \mathcal{S}\}$, and let u be its first singular vector
5: **for all** sentence s in $\mathcal{S}$ **do**
6: $\quad v_s \leftarrow v_s - uu^\top v_s$
7: **end for**

整体算法流程

3、实现

作者开源了其代码，地址为：

https://github.com/PrincetonML/SIF

Glove

1、概述

论文中作者总结了目前生成embedding的两大类方法，但这两种方法都有其弊端存在

- 基于矩阵分解，弊端为因为是基于全局进行矩阵的构建，对于一些高频词，在算法优化的过程中，占的权重比较大
- 基于滑动窗口，不能直接对语料库的单词进行共现建模，使用的是滑动窗口，没有办法利用数据的共现信息

因此作者提出了一种基于语料库进行信息统计，继而生成embedding的算法-Glove。下面就来具体看下对应的算法原理。

2、算法推导过程

字符的定义：

- X 表示单词的共现矩阵，X_{ij} 表示 单词j在单词i的上下文中出现的次数，即在指定的窗口内，单词i,j的共现次数
- X_i 表示 任何一个单词 k 和单词i的共现次数总和，$\sum_k X_{ik}$
- $P_{ij}=P(j|i)=X_{ij}/X_i$，表示单词i,j的共现次数在单词i 出现次数总和的概率

论文中给出了一个简单的例子，来数目如何从共现矩阵中提取出有意义的信息，如下图所示：

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

共现矩阵提取信息说明

上图想要说明的信息是，如果单词k 和单词i的相关度比k 和j的相关度大的话，$P(ik)/P(jk)$的值会很大，差距越大，比值越大；同理如果k和i的相关度比k和j的相关度小的话，$P(ik)/P(jk)$的值会很小，差距越大，比值越小；如果k和i,j都不想关的话，$P(ik)/P(jk)$的值会接近于1。

上述的讨论说明了词向量的学习应该是基于共现概率的比率而不是概率本身，因此假设可以通过函数F来学习到词向量，函数F可以抽象为：

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

其中w表示的是词向量，$\frac{P_{ik}}{P_{jk}}$ 可以从语料中计算得到，函数F依赖于一些尚未指定的参数，但因为一些必要的条件，函数F可以进一步的被确定。

由于向量空间具有线性结构，因此可以对词向量进行差分，函数F可以转化为：

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

虽然函数F可能是一个比较复杂的结构，比如神经网络，但这样做，会使我们试图捕获的线性结构消失，因此为了避免这个问题，我们可以先对向量做个内积，可以转化为：

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

上述公式中左侧是减法，右侧是除法，这很容易让人联想到指数运算，因此限定函数F为指数函数，此时有：

$$exp(w_i^T w_k - w_j^T w_k) = \frac{exp(w_i^T w_k)}{exp(w_j^T w_k)} = \frac{P_{ik}}{P_{jk}}$$

此时，只需要确保等式两边分子和分母相等即可，即：

$$exp(w_i^T w_k) = P_{ik}, exp(w_j^T w_k) = P_{jk}$$

进一步，可以转化为语料中的所有词汇，考察$exp(w_i^T w_k) = P_{ik} = \frac{X_{ik}}{X_i}$，即：

$$w_i^T w_k = log(\frac{X_{ik}}{X_i}) = logX_{ik} - logX_i$$

由于上式左侧$w_i^T w_k$ 中，调换i 和k 的值不会改变其结果，即具有对称性，因此，为了确保等式右侧也具备对称性，引入了两个偏置项，即：

$$w_i^T w_k = logX_{ik} - b_i - b_k$$

此时，$logX_i$ 已经包含在b_i中，因此，此时模型的目标就转化为通过学习词向量的表示，使得上式两边尽量接近，因此，可以通过计算两者之间的平方差来作为目标函数，即：

$$J = \sum_{i,k=1}^{V} (w_i^T \tilde{w}_k + b_i + b_k - logX_{ik})^2$$

但是这样的目标函数有一个缺点，就是对所有的共现词汇又是采用相同的权重，因此，作者对目标函数进行了进一步的修正，通过语料中的词汇共现统计信息来改变他们在目标函数中的权重，具体如下：

$$J = \sum_{i,k=1}^{V} f(X_{ik})(w_i^T \tilde{w}_k + b_i + b_k - logX_{ik})^2$$

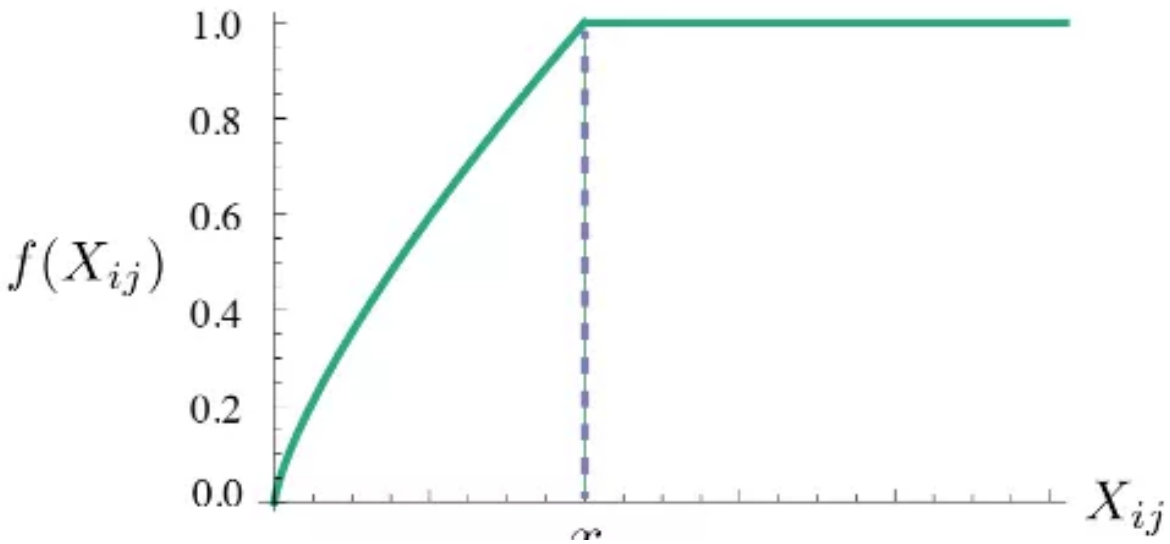
这里V 表示词汇的数量，并且权重函数f 必须具备一下的特性：

- $f(0) = 0$，当词汇共现的次数为0时，此时对应的权重应该为0
- $f(x)$ 必须时一个非减函数，这样才能保证当词汇共现的次数越大时，其权重不会出现下降的情况
- 对于那些太频繁的词，$f(x)$ 应该能给予他们一个相对小的数值，这样才不会出现过渡加权

综合以上三点特性，作者提出了下面的权重函数：

$$f(x) = \begin{cases} (x/x_{max})^a & if\, x < x_{max} \\ 1 & otherwise \end{cases}$$

作者在实验中设定$x_{max} = 100$，并且发现 $\alpha = 3/4$时效果比较好，函数的图像如下图所示：



x_{max}

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

a值设定效果

3、总结

以上就是有关$GloVe$原理的介绍，作者其实也是基于最开始的猜想一步一步简化模型的计算目标，最后看GloVe的目标函数时发现其实不难计算，但是要从最开始就想到这样一个目标函数其实还是很难的。最后做一下总结：

- $Glove$综合了全局词汇共现的统计信息和局部窗口上下文方法的优点，可以说是两个主流方法的一种综合，但是相比于全局矩阵分解方法，由于$GloVe$不需要计算那些共现次数为0的词汇，因此，可以极大的减少计算量和数据的存储空间
- 但是$GloVe$把语料中的词频共现次数作为词向量学习逼近的目标，当语料比较少时，有些词汇共现的次数可能比较少，笔者觉得可能会出现一种误导词向量训练方向的现象

4、实现

实现可以参考官方给出的代码和数据：

https://nlp.stanford.edu/projects/glove/

THE END