暑假学习总结

目录

一、Word2vec 模型

1.定义：首先，word2vec 是用于 word embedding 的模型，这些模型通常是用浅层（两层）神经网络训练的。其次，若单词上下文相似则训练得到的词向量在向量空间上接近。[1] 见下方

2.应用：机器翻译（比较直接）、推荐系统（Youtube）、作为 SVM 等模型的输入、情感分析

3.算法：

3.1 背景知识：

①Huffman 树与 Huffman 编码：word 是叶子节点，在语料中出现的次数是权值

②N-gram 模型：统计语言模型（参数太多）--->某词只和前面 n-1 个词相关（N-1 阶的 markou 假设）此时要注意平滑化。[2]

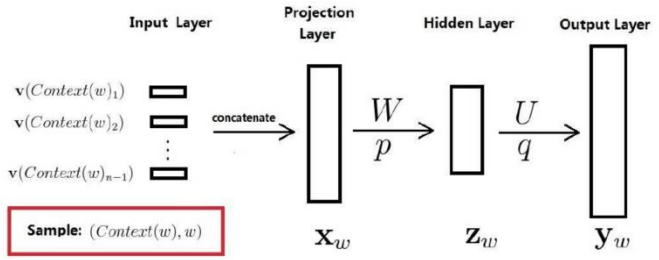
③概率模型函数：最大对数似然函数找最优参数 θ（上下文为 Context(w)时，word 是 w 的概率）

$$\mathcal{L}=\sum_{w\in C}\log p(w|Context(w))$$

将条件概率$p(w|Context(w))$视为关于w和$Context(w)$的函数

$$=\sum_{w\in C}\log F(w,Context(w),\theta)$$

④神经概率语言模型：词向量（One-hot Representation、Distributed Representation）、计算量太大（隐层到输出层的矩阵运算和输出层的归一化）



3.2word2vec 的优化部分：[6]

①输入层到投影层的运算改为叠加，即投影层节点数是节点个数。

②删去隐藏层，输入层是随机初始化的向量，输出层还是 y=∑w*x

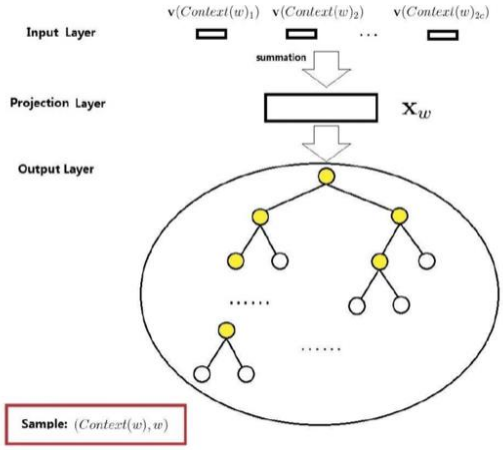
③优化 softmax：

Hieraichical Softmax：构造 Huffman 树，路径上的二分类概率连乘的结果就是该叶子节点对应的词的概率，不仅降低复杂度(log)，而且使高频词离根节点更近（训练速度快）

Negative Sampling（NCE 的简化版本）：提高训练速度，改善词质量，感觉只是为了最大化模型提出的随机负采样，已经不算是归一化了吧。（如何选取：测试发现最佳分布是 3/4 次幂的 unigram distribution，也就是随机选择，所以高频词选中几率更大）

4.两个模型

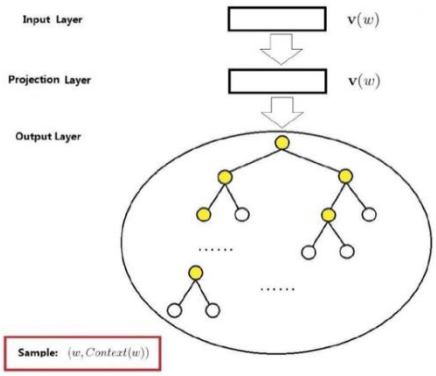
4.1 CBOW 模型(Continuous Bag-of-Words Model)：根据上下文预测当前词，最大化似然（梯度上升）



$$\begin{aligned}
&1.\ \mathbf{e}=0.\\
&2.\ \mathbf{x}_w=\sum_{u\in Context(w)}\mathbf{v}(u).\\
&3.\ \text{FOR } j=2:l^w \text{ DO}\\
&\quad\{\\
&\qquad 3.1\ \ q=\sigma(\mathbf{x}_w^\top\theta_{j-1}^w)\\
&\qquad 3.2\ \ g=\eta(1-d_j^w-q)\\
&\qquad 3.3\ \ \mathbf{e}:=\mathbf{e}+g\theta_{j-1}^w\\
&\qquad 3.4\ \ \theta_{j-1}^w:=\theta_{j-1}^w+g\mathbf{x}_w\\
&\quad\}\\
&4.\ \text{FOR } u\in Context(w) \text{ DO}\\
&\quad\{\\
&\qquad \mathbf{v}(u):=\mathbf{v}(u)+\mathbf{e}\\
&\quad\}
\end{aligned}$$

点击查看大图

4.2 Skip-gram 模型：根据当前词预测上下文，所以投影层没啥用吧。



$$\begin{aligned}
&\mathbf{e}=0\\
&\text{FOR } u\in Context(w) \text{ DO}\\
&\{\\
&\quad\text{FOR } j=2:l^u \text{ DO}\\
&\quad\{\\
&\qquad 1.\ \ q=\sigma(\mathbf{v}(w)^\top\theta_{j-1}^u)\\
&\qquad 2.\ \ g=\eta(1-d_j^u-q)\\
&\qquad 3.\ \ \mathbf{e}:=\mathbf{e}+g\theta_{j-1}^u\\
&\qquad 4.\ \ \theta_{j-1}^u:=\theta_{j-1}^u+g\mathbf{v}(w)\\
&\quad\}\\
&\}\\
&\mathbf{v}(w):=\mathbf{v}(w)+\mathbf{e}
\end{aligned}$$

点击查看大图

5.源码与应用：

源码：(github 上的 python 源码：gensim 库)

https://github.com/RaRe-Technologies/gensim/tree/develop/gensim/models

应用：首先 github 上面的是不能直接运行的，运行 deepwalk 的时候用到了 gensim==0.10.0 版本的 w2v。[10]

6. 问题和回答：

Q1：word2vec 模型中，为什么每个词会有一个 input vector 和一个 output vector？为什么要把 input vector 作为最终的输出结果？

A1：首先，当不用 Hieraichical Softmax 的时候，也就是正常的 softmax 时，输出层的权重是作为最终词向量的，因为每个词对应一个 y，对应一个权重向量，且维度是相同的。但是使用 Hieraichical 后，每个词输出的权重向量的维度不同（高频低维），长度不同肯定不能比较的呀。

Q2：判断两个词的含义相近，除了可以通过其上下文的相似性来判断，还可能有什么其他途径吗？这些途径可以用来继续扩展 word2vec 模型吗？

A2：好像很多是语言学的方法，有同义词词林（比较早），wordnet（“单词的网络”，好像还挺有意思的，对同义词词林扩展了，有各种关系），语义树（应用比较广泛），这些可以通过计算树状结构中相应节点的距离计算词语的相似度。而类似于 w2v 的这些方法是基于词语的相关性，即两个词如果都和另外一些词相关，则这两个词相关。

Q3：词本身也有很多特点：比如词频有高低之分，而且有的词含义单一，有的词有很多不同的含义。这些特点会在词向量上有所体现吗？

A3：词频高低是能体现出来的，但是一词多义这些没有体现。

Q4：在学习出的隐空间中，判断两个词的含义是否相似，可以利用两个词所对应的词向量的 cosine 距离衡量。用别的计算距离的方式可行吗？比如两个词在隐空间中的欧式距离。

A4：根据在信息检索课上学的知识来说，不行！比如把一个词复制一遍放在本来的词后面，这两个词含义是相同的，但是欧氏距离上却很大。

Q5：投影层是对词向量的叠加，这样就失去了词的顺序，如果改成首尾相连的方式，会有改进么？

A5：按道理来讲应该会提高对词义判断的准确性的，但是这个措施是为了提高模型的训练速度的，如果改成首尾相连，复杂度会变高，速度应该会减慢。

7. 补充知识：

NCE（噪声对比估计损失函数）：通过构造逻辑回归（logistic regression），对正样例和负样例做二分类，对于每一个样本，将自身的预测词 label 作为正样例，同时采样出 k 个其他词 label 作为负样例，从而只需要计算样本在这 k+1 个 label 上的概率。相比原始的 softmax 分类需要计算每个类别的分数，然后归一化得到概率，节约了大量的计算时间。

8. 文献和资料的简略说明：

[1] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in neural information processing systems. 2013: 3111-3119.

[2] http://blog.csdn.net/itplus/article/details/37969519

（word2vec 中的数学原理：对整个模型解释都比较清晰）

[3] Levy O, Goldberg Y. Neural word embedding as implicit matrix factorization[C]//Advances in neural information processing systems. 2014: 2177-2185.

(给 w2v 的解释，实质上模型相当于将原始矩阵分解成两个矩阵的乘积的形式）

[4] Rong X. word2vec parameter learning explained[J]. arXiv preprint arXiv:1411.2738, 2014.

[5] http://www.cnblogs.com/xlturing/p/6136690.html

[6] https://www.zybuluo.com/Dounm/note/591752

（Word2Vec-知其然知其所以然：也是一个梳理的博客，思路清晰，可以配合【2】中的博客看）

[7] http://blog.csdn.net/mmc2015/article/details/50943163

（各种词语相似度的计算）

[8] http://superjom.duapp.com/neural-language-model/word2vec-implement.html

（word2vec 代码实现：gensim 库的源码，有注释）

[9] 刘群, 李素建. 基于《知网》的词汇语义相似度计算[J]. 中文计算语言学, 2002, 7(2): 59-76.

（里面的一些词语相似度比较的方法不错，从这里看到的语义知识树）

[10] http://www.10tiao.com/html/506/201704/2651615429/1.html

（Word2vec 实战：数据集代码都有）

二、网络表示学习

1. 理解：对网络结构的数据挖掘，对网络数据进行合理表示来输入到机器学习算法中。[11]（network embedding）

2. 应用：节点标签预测、网络信息传播预测、知识图谱学习表示、推荐系统

3. 算法类别：（陈维政的网络表示学习简略版）

2.1 谱算法：利用特征值和特征向量，奇异值和奇异向量

PCA（或 SVD）——样本的协方差矩阵的特征向量进行降维（只考虑了结构信息）

LLE（改进版 LaplacianEigenmaps）——近邻点的线性加权组合的非线性降维算法。（只能处理无向网络）

DGE——基于随机游走的思想。

2.2 最优化：通过最大化或者最小化一个明确的优化目标函数，求得节点的低维表示。

LSHM——同时学习节点的向量表示和标签的线性分类函数（可以处理异构网络，把节点属性扩展为一种没有标签的节点）

2.3 概率生成式模型：基于概率的生成的采样过程去建模网络数据的产生过程，输入一般是文本网络。

Link-PLSA-LDA:为学术论文引用网络设计的。

RTM：建模了链接关系。

PLANE：对 RTM 扩展，用 PE 算法对文章话题降维。

2.4 力导向：（没看过）

2.5 深度学习：（重点看的 3 个算法下面详细说）

4.Deepwalk 算法：

4.1 对论文的理解：首先，感觉这篇论文的亮点在于作者观察到文本语料中词语出现的频率和随机游走的网络上节点被访问到的次数都服从幂律分布（zipf），这成为 deepwalk 成立的前提。所以可以把节点作为单词，随机游走的路径作为句子，输入到 word2vec 中训练得到节点的向量表示。所以缺点应该是没有明确的优化目标函数，所以给后面人提供很大改进空间。

4.2 github 源码：

https://github.com/phanein/deepwalk

数据集定义：

http://leitang.net/social_dimension.html

4.3 算法理解和应用：

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

主要步骤：Network/graph---random walk---得到节点序列（representation mapping）---放到 skip-gram 模型中（word2vec 模型）----output：representation

配置与运行：（放在自己 csdn 的博客上面了：http://blog.csdn.net/qq_26919935/article/details/76574845）

5.Line 算法：

5.1 对论文的理解：作者认为网络中存在两种 proximity，一个是“first-order proximity”，即 a,b 两个节点有边直接相连，则为 W(a,b)，另一个是“second-order proximity”，即 a,b 之间共同好友集合的 proximity，然后 Line 算法分别设计优化目标函数，并以两种向量的拼接作为作为节点的最终向量表示。利用了 KL 散度来保持 NRL 前后节点之间相似度不变（deepwalk 只考虑了 second-order proximity）

5.2 补充知识：KL 散度从直观上是个度量或距离函数，但它并不是一个真正的度量或者距离，因为它不具有对称性。[20]

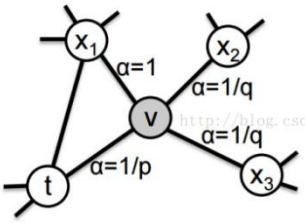
6.Node2vec 算法：

6.1 论文理解：和 deepwalk 最大的不同是它定义了两个参数 p 和 q，在 BFS 和 DFS 中达到一个平衡（BFS 代表微观局部信息，DFS 代表宏观全局信息）本质上相当于对当前节点的邻域节点的采样，同时保留了该节点在网络中的位置信息。

参数控制跳转概率的随机游走，之前完全随机时，p=q=1.

--返回概率参数 p，对应 BFS，p 控制回到原来节点的概率，如图中从 t 跳到 v 以后，有 1/p 的概率在节点 v 处再跳回到 t。

--离开概率参数 q，对应 DFS，q 控制跳到其他节点的概率。



$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

6.2 伪代码：

Algorithm 1 The *node2vec* algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)
  π = PreprocessModifiedWeights(G, p, q)
  $G' = (V, E, \pi)$
  Initialize *walks* to Empty
  for *iter* = 1 **to** r **do**
    for all nodes $u \in V$ **do**
      $walk$ = node2vecWalk(G', u, l)
      Append $walk$ to *walks*
  f = StochasticGradientDescent($k, d,$ *walks*)
  return f

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u, Length l)
  Inititalize $walk$ to [u]
  for *walk_iter* = 1 **to** l **do**
    $curr = walk[-1]$
    V_{curr} = GetNeighbors($curr, G'$)
    s = AliasSample(V_{curr}, π)
    Append s to $walk$
  return $walk$

6.3 扩展：广告推荐和 Youtube 视频推荐（已经实现并且有源码）

7.问题与回答：

Q1:这些 embedding 的算法套上 w2v 总感觉很牵强，

能不能想一些网络结构自己的生成向量表示的方式呢？

Q2：一个网络的结构和该网络上的随机游走有什么联系吗？或者说网络的某些结构特征在随机游走序列上会有什么反映吗？

A2：能想到的能体现出的结构特征比如①显然与节点有连接的边的节点可以访问得到，对邻域节点的采样，同时保留了节点位置信息。②周围节点的访问频率符合幂律，入度高（directed graph）的节点随机访问的次数相对多。

Q3：网络与文本的相似之处在哪里？不同之处又有哪些？

A3：相似之处比如①随机游走时相似节点的游走序列和相似，文本中词语相似的上下文也相似。②如果两个节点相似性越高，则两个节点存在链接或者未来形成链接的可能性越高，这样的节点可以构成上下文。③异构网络可不可以理解成文本中的一词多义的情况？

不同之处如网络中的节点可能有节点自身属性，比如爬取的 url 节点属性可以是网页内容，但是文本的单词没有太长的属性（但是有单词的解释，不知道算不算）

Q4：判断两个点是否相似，除了通过其邻接节点的相似性来判断，还有其他途径吗？可以用来继续扩展 DeepWalk 吗？

A4：Line 中提出的两个节点相似度，一个是直接相连的节点相似，一个是间接相连节点相似（题目中的意思）那么能不能从节点自身带的文本（节点属性）去判断呢？比如 url 节点附带的网页内容的相似度去判断，是不是相当于语义上相似度判断，比较复杂？

但是觉得从邻接节点相似度去判断是很有道理的，比如 jaccard 系数，余弦相似度都是利用的邻接节点。

8. 文献和资料的简略说明：

[11]陈维政, 张岩, 李晓明. 网络表示学习[J]. 大数据, 2017, 1(3): 2015025.

[12]Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations[C]//Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014: 701-710.

[13]http://blog.csdn.net/wolfblood_zzx/article/details/73088111

[14]http://www.perozzi.net/

[15]http://www.cnblogs.com/lavi/p/4323691.html
（对代码的简单介绍）

[16]http://blog.csdn.net/pipisorry/article/details/52098864
（非负矩阵分解（NMF,Non-negative matrix factorization））

[17]http://blog.csdn.net/u013527419/article/details/76017528
（网络表示学习的综述，简略但是清晰的讲了 dw,line 和 node2vec3 个算法）

[18]Tang J, Qu M, Wang M, et al. Line: Large-scale information network embedding[C]//Proceedings of the 24th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2015: 1067-1077.
（Line 算法论文）

[19]Grover A, Leskovec J. node2vec: Scalable feature learning for networks[C]//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2016: 855-864
（node2vec 论文）

[20]http://blog.csdn.net/u013527419/article/details/51776786
（KL 散度的介绍）.

[21]http://snap.stanford.edu/node2vec/
（斯坦福的介绍，包括了介绍和数据集、实现等等）

[22]http://geek.csdn.net/news/detail/200138
（深度学习在推荐领域的应用，讲的是 node2vec 在广告推荐上的应用）

[23]https://www.zhihu.com/question/57269332
（知乎：有哪些指标可以描述两个图（graph）的相似度？）

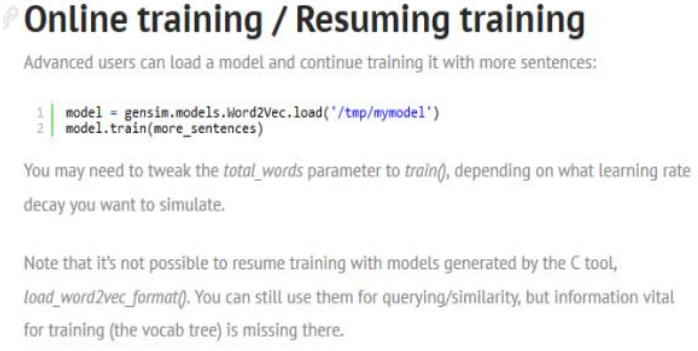
[24]http://www.sohu.com/a/154650431_642762
（很全的 network embedding 的总结,刚发现的很不错）

[25]https://zhuanlan.zhihu.com/p/24769965
（《Structural Deep Network Embedding》阅读笔记）

三、增量式学习(incremental learning)

1. 对目前研究的概述：在文本语料的处理方面，word2vec 模型有效且适用于机器学习。网络的 embedding 比如 deepwalk,line,node2vec 比较出名。但是这些都是静态方法，需要把整个语料和网络输入进去。我看到的今年两篇论文以及相关的讨论都是对 w2v 等词向量的增量式改进，很少对网络的增量式表示学习，所以可以根据网络动态演化规律来提出一种增量式网络表示学习方法，并通过理论分析和实验具体验证有效性。

2. gensim 包对数据量很大时的处理：逐条的读取数据。



Gensim only requires that the input must provide sentences *sequentially*, when iterated over. No need to keep everything in RAM: we can provide one sentence, process it, forget it, load another sentence...

For example, if our input is strewn across several files on disk, with one sentence per line, then instead of loading everything into an in-memory list, we can process the input file by file, line by line:

```
class MySentences(object):
    def __init__(self, dirname):
        self.dirname = dirname

    def __iter__(self):
        for fname in os.listdir(self.dirname):
            for line in open(os.path.join(self.dirname, fname
                yield line.split()

sentences = MySentences('/some/directory') # a memory-friend
model = gensim.models.Word2Vec(sentences)
```

3. 词向量模型的增量式学习方法（ILW）[24]：对 Word2vec 中 CBOW 模型的 NS 算法进行改进，主要步骤就是——对新增文本中出现的新词进行初始化更新，然后基于历史词表的反例采样。也就是遍历一遍新加入的语料中所有词，无论是否是“新词”，都是“不去重”的加入到词表（word-list），然后再重新去遍历新语料去 NS。那这样的话和静态 w2v 又有什么区别呢？并没有用到原始语料已经学习出的特点，应该不是很好的增量式模型。

4. 可行但是有局限性的方法：首先训练好一个语料库，然后比如搜索“北京大学”，但是这个语料库里面没有这个词，就可以去爬取所有和“北京大学”相关的网页，提取相关句子，词的上下文假如为 a,b,c,d，就是已知 a,b,c,d 去求“北京大学”出现的概率，最大化这个概率就可以。但是如果语料库过小，不包括 a,b,c,d 就会滚雪球了，所以有局限性。改进方法：把已知的和未知的分开，未知的去“查词典”或者猜测或者直接扔掉处理。

5. 关于 word2vec 对分词方面的缺陷处理：例如如果要把 word2vec 用在新闻稿上头的话就有极大的问题，因为新闻稿里头的新词层出不穷，会强迫我必须不停的重新做分词以及重新训练 word2vec 模型，故完全不实用。所有可不可以分层训练词语，比如先训练“北京大学”，然后训练“北京”、“大学”分别的词向量，然后加权平均得到最终词向量。

6. github 上增量 word2vec 模型源码：https://github.com/wammar/incremental-word2vec

Algorithm 1 Incremental SGNS

$f(w) \leftarrow 0$ for all $w \in \mathcal{W}$
for $i = 1, \ldots, n$ **do**
  $f(w_i) \leftarrow f(w_i) + 1$
  $q(w) \leftarrow \frac{f(w)^\alpha}{\sum_{w' \in \mathcal{W}} f(w')^\alpha}$ for all $w \in \mathcal{W}$
  for $j = -c, \ldots, -1, 1, \ldots, c$ **do**
    draw k negative samples from $q(w)$: $v_1, \ldots, v_k$
    use SGD to update $\mathbf{t}_{w_i}$, $\mathbf{c}_{w_{i+j}}$, and $\mathbf{c}_{v_1}, \ldots, \mathbf{c}_{v_k}$
  end for
end for

Algorithm 2 Mini-batch SGNS

for each subset $\mathcal{D}$ of the training data **do**
  update the noise distribution using $\mathcal{D}$
  perform SGD over $\mathcal{D}$
end for

Algorithm 3 Adaptive unigram table.

$f(w) \leftarrow 0$ for all $w \in \mathcal{W}$
$z \leftarrow 0$
for $i = 1, \ldots, n$ **do**
  $f(w_i) \leftarrow f(w_i) + 1$
  $F \leftarrow f(w_i)^\alpha - (f(w_i) - 1)^\alpha$
  $z \leftarrow z + F$
  if $|T| < \tau$ **then**
    add F copies of w_i to T
  else
    for $j = 1, \ldots, \tau$ **do**
      $T[j] \leftarrow w_i$ with probability $\frac{F}{z}$
    end for
  end if
end for

8. 问题与解答：

Q1：word2vec 扩展成增量式模型的主要难点在哪里？或者说 word2vec 这个模型究竟适不适合扩展成增量式模型呢。

A1：就像上面提到的，难点比如分词方面会有错误，其次，若是增量很多，模型泛化能力应该会下降的。现在看到的增量模型都是对 w2v 的简单修正，没有用上太多原来训练模型得到的参数，生成式的模型不容易。

Q2：在什么场景下，文本材料也是动态的？它们是怎样实现增量式学习的？也许可以借鉴一下其中的思想。

A2：比如新闻稿的发布，网络上不断有贴吧的文本发布等。

Q3：网络的动态性可以体现在以下两方面，在考虑增量模型时，是否要分情况讨论呢？

- 动态的边：原来有的边可能会消失，原来没有的边也可能出现（假设节点数目不变）。
- 网络是不断扩张的：有新的点加入，且边也有更新。

A3：其实更看重数据集吧，有的数据集边的情况是比较固定的，只有当新点加入时候才会有边的更新（就像是周一讲的美国两党的数据集）若是像人参加俱乐部这种数据集，边和点都非常活跃，需要一起讨论的。

9. 文献和资料的简略说明：

[26]http://www.vipzhuanli.com/pat/books/201610995636.3/2.html

[27]http://www.vipzhuanli.com/pat/books/201710022618.1/2.html

[28]http://tieba.baidu.com/p/4732041271

[29]Kaji N, Kobayashi H. Incremental skip-gram model with negative sampling[J]. arXiv preprint arXiv:1704.03956, 2017.

（对 w2v 的 SGNS 的改进）

[30]Peng H, Li J, Song Y, et al. Incrementally Learning the Hierarchical Softmax Function for Neural Language Models[C]//AAAI. 2017: 3267-3273.

（对 w2v 的 Hierarchical Softmax 模型改进）