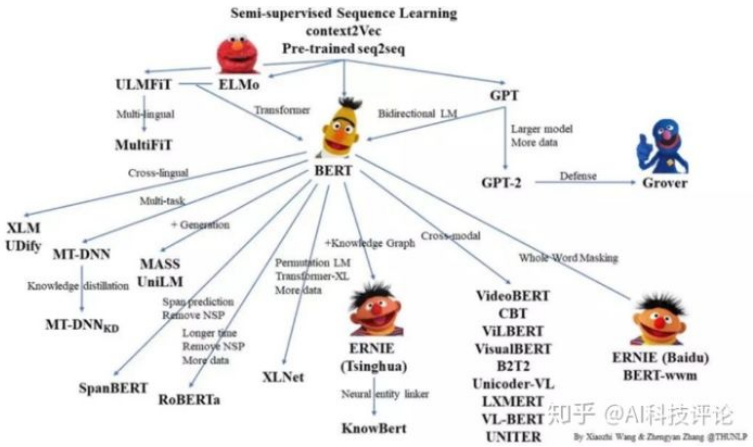
NLP学习算法3

- TF-idf



  - one-hot 篇
    - 1.1 为什么有 one-hot ?
      - 由于计算机无法识别 文本语言，所以需要将文本数字化，one-hot 方法最早的一种将 文本数字化的方法。
    - 1.2 one-hot 是什么?
      - 用一个很长的向量来表示一个词，向量长度为词典的大小N，每个向量只有一个维度为1，其余维度全部为0，为1的位置表示该词语在词典的位置。
    - 1.3 one-hot 有什么特点?
      - 维度长：向量的维度为 **词典大小**；
      - ——其零：每个向量**只有一个维度为1**，其余维度全部为0，**为1的位置表示该词语在词典的位置**；
    - 1.4 one-hot 存在哪些问题?
      - **维度灾难**：容易受维数灾难的困扰，每个词语的维度就是语料库字典的长度；
      - **离散、稀疏问题**：因为 one-Hot 中，句子向量，如果词出现则为1，没出现则为0，但是由于维度远大于句子长度，所以句子中的1远小于0的个数；
      - **维度鸿沟问题**：词语的编码往往是随机的，导致不能很好地刻画词与词之间的相似性。

- TF-IDF 篇
  - 2.1 什么是 TF-IDF?
    - TF-IDF 是一种统计方法，用以评估句子中的某一个词（字）对于整个文档的重要程度。
  - 2.2 TF-IDF 如何评估词的重要程度?
    - 对于 句子中的某一个词（字）随着其在**整个句子**中的出现次数的增加，其重要性也随着**增加**；（正比关系）【体现词在句子中频繁性】
    - 对于 句子中的某一个词（字）随着其在**整个文档**中的出现频率的增加，其重要性也随着**减少**；（反比关系）【体现词在文档中的唯一性】
  - 2.3 TF-IDF 的思想是什么?
    - 如果某个单词在一篇文章中出现的频率TF高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类；
  - 2.4 TF-IDF 的计算公式是什么?
    - **词频 （Term Frequency，TF）**
      - 介绍：体现 词 在 句子 中出现的频率；
      - 问题：
        - 当一个句子长度的增加，句子中 每一个 出现的次数也会随之增加，导致该值容易偏向长句子；
        - 解决方法：
          - 需要做归一化（词频除以句子总字数）
      - 公式

$$TF_w = \frac{\text{在某一类中词条}w\text{出现的次数}}{\text{该类中所有的词条数目}}$$

    - **逆文本频率(Inverse Document Frequency，IDF)**
      - 介绍：体现 词 在文档 中出现的频率
      - 方式：某一特定词语的IDF，可以由总句子数目除以包含该词语的句子的数目，再将得到的商取对数得到；

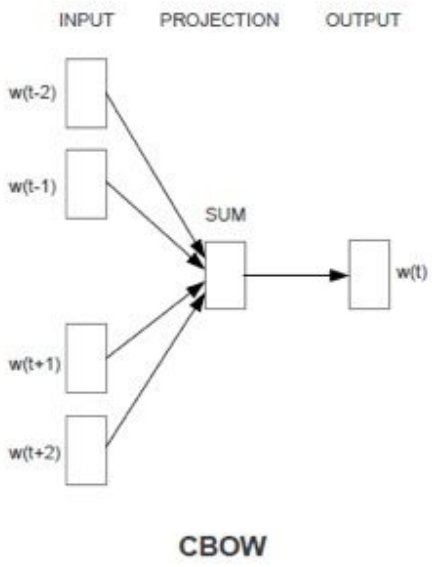
- 作用：如果包含词条t的文档越少，IDF越大，则说明词条具有很好的类别区分能力
- 公式：

$$IDF = log(\frac{\text{语料库的文档总数}}{\text{包含词条}w\text{的文档数}+1}), \text{分母之所以要加1，是为了避免分母为0}$$

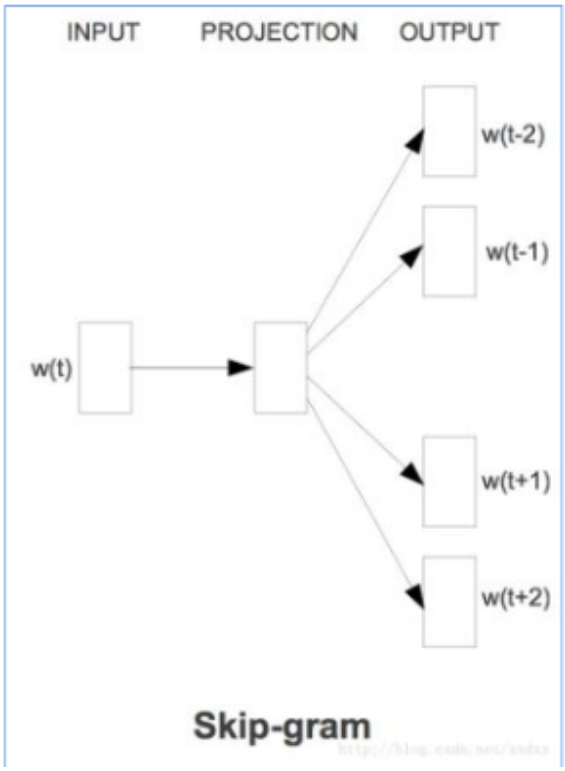
- 2.5 TF-IDF 怎么描述?
  - 某一特定句子内的高词语频率，以及该词语在整个文档集合中的低文档频率，可以产生出高权重的TF-IDF。因此，TF-IDF倾向于过滤掉常见的词语，保留重要的词语。
- 2.6 TF-IDF 的优点是什么?
  - 容易理解；
  - 容易实现；
- 2.7 TF-IDF 的缺点是什么?
  - 其简单结构并没有考虑词语的语义信息，无法处理一词多义与一义多词的情况。
- 2.8 TF-IDF 的应用?
  - 搜索引擎；
  - 关键词提取；
  - 文本相似性；
  - 文本摘要

- Word2vec
  - Word2vec 介绍篇
    - 1.1 Word2vec 指什么?
      - 介绍：word2vec是一个把词语转化为对应向量的形式。word2vec中建模并不是最终的目的，其目的是获取建模的参数，这个过程称为fake task。
      - 双剑客
        - CBOW vs Skip-gram
    - 1.2 Wordvec 中 CBOW 指什么?
    - **CBOW**
      - 思想：用周围词预测中心词

- 输入输出介绍：输入是某一个特征词的上下文相关的词对应的词向量，而输出就是这特定的一个词的词向量



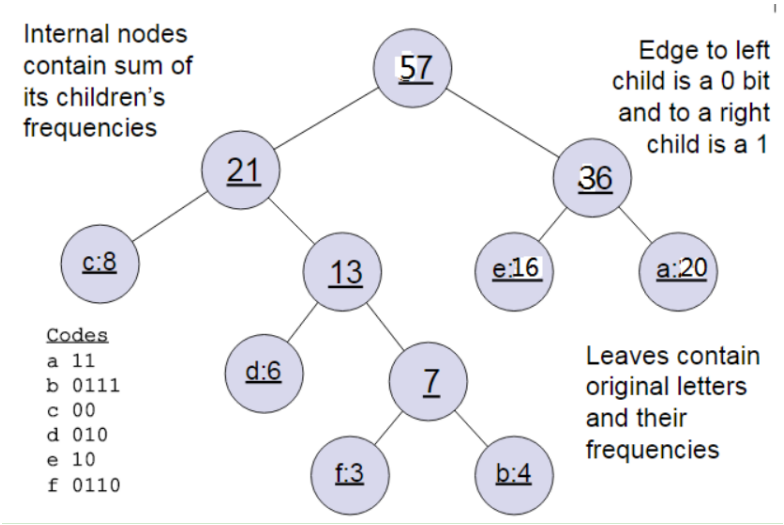
- 1.3 Wordvec 中 Skip-gram 指什么?
- **Skip-gram**
  - 思想：用中心词预测周围词
  - 输入输出介绍：输入是特定的一个词的词向量，而输出是特定词对应的上下文词向量



- 1.4 CBOW vs Skip-gram 哪一个好?

- CBOW 可以理解为 一个老师教多个学生；（高等教育）
- Skip-gram 可以理解为 一个学生被多个老师教；（补习班）
- 那问题来了？
  - 最后 哪个学生 成绩 会更好？

- Word2vec 优化篇
  - 2.1 Word2vec 中 **霍夫曼树** 是什么？
    - HS用哈夫曼树，把预测one-hot编码改成预测一组01编码，进行层次分类。
    - 输入输出：
      - 输入：权值为(w1,w2,...wn)的n个节点
      - 输出：对应的霍夫曼树
    - 步骤：
      - 将(w1,w2,...wn)看做是有n棵树的森林，每个树仅有一个节点。
      - 在森林中选择根节点权值最小的两棵树进行合并，得到一个新的树，这两颗树分布作为新树的左右子树。新树的根节点权重为左右子树的根节点权重之和。
      - 将之前的根节点权值最小的两棵树从森林删除，并把新树加入森林。
      - 重复步骤2）和3）直到森林里只有一棵树为止。
    - 举例说明：下面我们用一个具体的例子来说明霍夫曼树建立的过程
      - 有(a,b,c,d,e,f)**共6个节点，节点的权值分布是(20,4,8,6,16,3)。**
      - 首先是最小的b和f合并，得到的新树根节点权重是7.此时森林里5棵树，根节点权重分别是20,8,6,16,7。此时根节点权重最小的6,7合并，得到新子树，依次类推，最终得到下面的霍夫曼树。



- 2.2 Word2vec 中 为什么要使用 霍夫曼树?
  - 一般得到霍夫曼树后我们会对叶子节点进行霍夫曼编码，由于权重高的叶子节点越靠近根节点，而权重低的叶子节点会远离根节点，这样我们的高权重节点编码值较短，而低权重值编码值较长。这保证的树的带权路径最短，也符合我们的信息论，即我们希望越常用的词拥有更短的编码。如何编码呢？一般对于一个霍夫曼树的节点（根节点除外），可以约定左子树编码为0，右子树编码为1.如上图，则可以得到c的编码是00。
  - 在word2vec中，约定编码方式和上面的例子相反，即约定左子树编码为1，右子树编码为0，同时约定左子树的权重不小于右子树的权重。
- 2.3 Word2vec 中使用 霍夫曼树 的好处?
  - 由于是二叉树，之前计算量为V,现在变成了log2V;
  - 由于使用霍夫曼树是高频的词靠近树根，这样高频词需要更少的时间会被找到，这符合我们的贪心优化思想。
- 2.4 为什么 Word2vec 中会用到 负采样?
  - 动机：使用霍夫曼树来代替传统的神经网络，可以提高模型训练的效率。但是如果我们的训练样本里的中心词w是一个很生僻的词，那么就得在霍夫曼树中辛苦的向下走很久了;
  - 介绍：一种概率采样的方式，可以根据词频进行随机抽样，倾向于选择词频较大的负样本;
  - 例子:

- 比如给定一句话“这是去上学的班车”，则对这句话进行正采样，得到上下文“上”和目标词“学”，则这两个字就是正样本。负样本的采样需要选定同样的“上”，然后在训练的字典中任意取另一个字，“梦”、“目”，这一对就构成负样本。训练需要正样本和负样本同时存在。

- 优点：
  - 用来提高训练速度并且改善所得到词向量的质量的一种方法；
  - 不同于原本每个训练样本更新所有的权重，负采样每次让一个训练样本仅仅更新一小部分的权重，这样就会降低梯度下降过程中的计算量。

- 2.5 Word2vec 中会用到 负采样 是什么样?
  - 因为使用softmax时，分母需要将中心词与语料库总所有词做点乘，代价太大：所以负采样方法将softmax函数换成sigmoid函数。

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

  - 选取K个负样本，即窗口之外的样本，计算中心词与负样本的点乘，最小化该结果。计算中心词与窗口内单词的点乘，最大化该结果，目标函数为：

$$J_t(\theta) = \log \sigma\left(u_o^T v_c\right) + \sum_{j \sim P(w)}^{k} \mathbb{E}\left[\log \sigma\left(-u_j^T v_c\right)\right]$$

- 2.6 Word2vec 中 负采样 的采样方式?
  - NS是一种概率采样的方式，可以根据词频进行随机抽样，我们倾向于选择词频比较大的负样本，比如“的”，这种词语其实是对我们的目标单词没有很大贡献的。
  - Word2vec则在词频基础上取了0.75次幂，减小词频之间差异过大所带来的影响，使得词频比较小的负样本也有机会被采到。

- 极大化正样本出现的概率，同时极小化负样本出现的概率，以sigmoid来代替softmax，相当于进行二分类，判断这个样本到底是不是正样本。

- 对比篇
  - 3.1 word2vec和NNLM对比有什么区别？（word2vec vs NNLM）
    - NNLM：是神经网络语言模型，使用前 n - 1 个单词预测第 n 个单词;
    - word2vec：使用第 n - 1 个单词预测第 n 个单词的神经网络模型。但是 word2vec 更专注于它的中间产物词向量，所以在计算上做了大量的优化。优化如下：
      - 对输入的词向量直接按列求和，再按列求平均。这样的话，输入的多个词向量就变成了一个词向量。
      - 采用分层的 softmax(hierarchical softmax)，实质上是一棵哈夫曼树。
      - 采用负采样，从所有的单词中采样出指定数量的单词，而不需要使用全部的单词
  - 3.2 word2vec和tf-idf 在相似度计算时的区别?
    - word2vec 是稠密的向量，而 tf-idf 则是稀疏的向量；
    - word2vec 的向量维度一般远比 tf-idf 的向量维度小得多，故而在计算时更快；
    - word2vec 的向量可以表达语义信息，但是 tf-idf 的向量不可以；
    - word2vec 可以通过计算余弦相似度来得出两个向量的相似度，但是 tf-idf 不可以；
- word2vec 实战篇
  - 4.1 word2vec训练trick，window设置多大?
    - window（窗口）设置：
      - 比较大，会提取更多的topic信息
      - 设置比较小的话会更加关注于词本身。
    - 默认参数是5，但是在有些任务中window为2效果最好，比如某些英语语料的短文本任务（并非越大越好）

- 4.1 word2vec训练trick，词向量纬度，大与小有什么影响，还有其他参数?
  - 词向量维度代表了词语的特征，特征越多能够更准确的将词与词区分，就好像一个人特征越多越容易与他人区分开来。但是在实际应用中维度太多训练出来的模型会越大，虽然维度越多能够更好区分，但是词与词之间的关系也就会被淡化，这与我们训练词向量的目的是相反的，我们训练词向量是希望能够通过统计来找出词与词之间的联系，维度太高了会淡化词之间的关系，但是维度太低了又不能将词区分，所以词向量的维度选择依赖于你的实际应用场景，这样才能继续后面的工作。一般说来200-400维是比较常见的。

- fastText
  - fastText 动机篇
    - 1.1 word-level Model 是什么?
      - 介绍：基于word单词作为基本单位的，这种方式虽然能够很好的对词库中每一个词进行向量表示
    - 1.2 word-level Model 存在什么问题?
      - OOV 问题
        - 问题描述：容易出现单词不存在于词汇库中的情况;
        - 解决方法：最佳语料规模，使系统能够获得更多的词汇量;
      - 误拼障碍
        - 问题描述：如果遇到了不正式的拼写, 系统很难进行处理;
        - 解决方法：矫正或加规则约束;
      - 做翻译问题时, 音译姓名比较难做到
    - 1.3 Character-Level Model 是什么?
      - 介绍：基于 Character 作为基本单位的，这种方式虽然能够很好的对字库中每一个 Char 进行向量表示
    - 1.4 Character-Level Model 优点?

- 能够解决 Word-level 所存在的 OOV 问题；
- 拼写类似的单词 具有类似的 embedding；

- 1.5 Character-Level Model 存在问题?
  - Character-level 的输入句子变长；
  - 数据变得稀疏；
  - 对于远距离的依赖难以学到；
  - 训练速度降低；
- 1.6 Character-Level Model 问题的解决方法?
  - Lee 等 提出了利用多层 conv(卷积) 和 pooling 和 highway layer 的方式来解决该问题，其结构如下所示：
    - 输入的字符首先需要经过 Character embedding 层，并被转化为 character embeddings 表示；
    - 采用 不同窗口大小的卷积核对输入字符的 character embeddings 表示进行卷积操作，论文中采用的窗口的大小分别为 3、4、5 ，也就是说学习 Character-level 的 3-gram、4-gram、5-gram;
    - 对不同卷积层的卷积结果进行 max-pooling 操作，即捕获其最显著特征生成 segment embedding;
    - segment embedding 经过 Highway Network (有些类似于Residual network，方便深层网络中信息的流通，不过加入了一些控制信息流量的gate）；
    - 输出结果 再经过 单层 BiGRU，得到最终 的 encoder output;
    - 之后，decoder再利用Attention机制以及character level GRU进行decode
  - 通过这种方式不仅能够解决 Word-level 所存在的 OOV 问题，而且能够捕获 句子的 3-gram、4-gram、5-gram 信息，这个也是 后期 FastText 的想法雏形；

- 词内的n-gram信息(subword n-gram information) 介绍篇
  - 2.1 引言

- 在前面，我们已经介绍和比较了 word-level 和 character-level 的优缺点，并根据其特点，提出一种介于 word-level Model 和 Character-level 之间的 Model —— Subword Model。
- 那么，我们可不可以采取类似于上面的subword的思路来产生更好的word embedding呢?
- FAIR的FastText就是利用subword将word2vec扩充，有效的构建embedding。

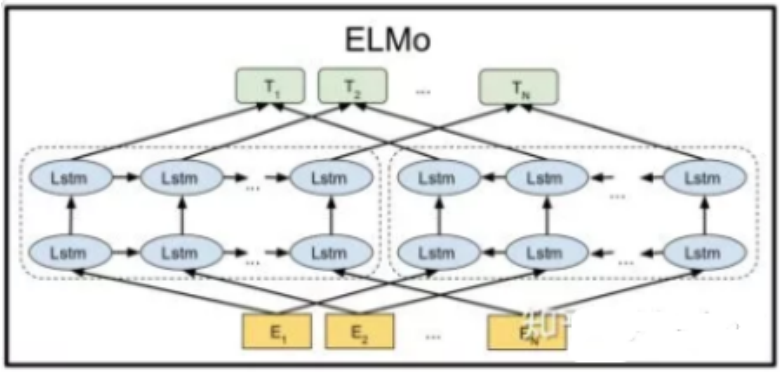
- 2.2 fastText 是什么?
  - 将每个 word 表示成 bag of character n-gram 以及单词本身的集合，例如对于where这个单词和n=3的情况，它可以表示为 <wh,whe,her,ere,re>, ，其中"<",">"为代表单词开始与结束的特殊标记。
  - 假设对于word w ，其n-gram集合用 G_w 表示，每个 n-gram 的矢量表示为,则每个单词可以表示成其所有n-gram的矢量和的形式，而center word w 与context word c 的分数就可表示成
  - 之后就可以按照经典的word2vec算法训练得到这些特征向量。
  - 这种方式既保持了word2vec计算速度快的优点，又解决了遇到training data中没见过的oov word的表示问题，可谓一举两得。
- 2.3 fastText 的结构是什么样?
  - 每个单词通过嵌入层可以得到词向量;
  - 然后将所有词向量平均可以得到文本的向量表达;
  - 在输入分类器，使用softmax计算各个类别的概率;
- 2.4 为什么 fastText 要使用词内的n-gram信息(subword n-gram information)?
  - 之前方法:
    - 以词汇表中的独立单词作为基本单元来进行训练学习的
    - 存在问题:

        - 低频词、罕见词：由于在语料中本身出现的次数就少，得不到足够的训练，效果不佳
        - 未登录词：如果出现了一些在词典中都没有出现过的词，或者带有某些拼写错误的词，传统模型更加无能为力
- 2.5 fastText 词内的n-gram信息(subword n-gram information) 介绍?
    - s1. 将一个单词打散到字符级别；
    - s2. 利用字符级别的n-gram信息来捕捉字符间的顺序关系
        - 目的：以此丰富单词内部更细微的语义
    - 举例：
        - 对于一个单词“google”，为了表达单词前后边界，我们加入<>两个字符，即变形为“”；
        - 抽取所有的tri-gram信息：G = { <go, goo, oog,ogl, gle, le>}；
        - 通过这种方式：原始的一个单词google，就被一个字符级别的n-gram集合所表达；
- 2.6 fastText 词内的n-gram信息 的 训练过程?
    - s1:每个n-gram都会对应训练一个向量；
    - s2:原来完整单词的词向量就由它对应的所有n-gram的向量求和得到；
    - s3:所有的单词向量以及字符级别的n-gram向量会同时相加求平均作为训练模型的输入；
- 2.7 fastText 词内的n-gram信息 存在问题?
    - 由于需要估计的参数多，模型可能会比较膨胀
    - 压缩模型的建议：
        - 采用hash-trick：由于n-gram原始的空间太大，可以用某种hash函数将其映射到固定大小的buckets中去，从而实现内存可控；
        - 采用quantize命令：对生成的模型进行参数量化和压缩；
        - 减小最终向量的维度。

- 层次化Softmax回归(Hierarchical Softmax) 介绍篇
  - 3.1 为什么要用 层次化Softmax回归(Hierarchical Softmax) ?
    - 传统 softmax
      - 介绍：
        - 以隐藏层的输出h为输入，经过线性和指数变换后，再进行全局的归一化处理，找到概率最大的输出项；
      - 问题：
        - 当词汇数量V较大时（一般会到几十万量级），Softmax计算代价很大，是O(V)量级。
  - 3.2 层次化Softmax回归(Hierarchical Softmax) 的思想是什么?
    - 将一个全局多分类的问题，转化成为了若干个二元分类问题，从而将计算复杂度从O(V)降到O(logV)；
    - 每个二元分类问题，由一个基本的逻辑回归单元来实现
  - 3.3 层次化Softmax回归(Hierarchical Softmax) 的步骤?
    - 步骤：
      - 从根结点开始，每个中间结点（标记成灰色）都是一个逻辑回归单元，根据它的输出来选择下一步是向左走还是向右走；
      - 上图示例中实际上走了一条“左-左-右”的路线，从而找到单词w_2。而最终输出单词w_2的概率，等于中间若干逻辑回归单元输出概率的连乘积；
- fastText 存在问题?
  - 如何构造每个逻辑回归单元的输入
    - 特殊函数 $[\![x]\!]$
      - 如果下一步需要向左走其函数值定义为1，向右则取-1。在训练时，我们知道最终输出叶子结点，并且从根结点到叶子结点的每一步的路径也是确定的。
    - 每个内部结点（逻辑回归单元）对应的一个向量 v'

- 以在训练过程中学习和更新
- h 是网络中隐藏层的输出
- 如何建立这棵用于判断的树形结构?
- 霍夫曼树的构造
  - 处理机制：将字符信息编码成为0/1二进制串
  - 结构介绍：给出现频繁的字符较短的编码，出现较少的字符以较长的编码，是最经济的方案
  - 构造步骤：

- Elmo
  - Elmo 动机篇
    - 1.1 为什么会有 Elmo?
      - 多义词问题：
        - 因为 one-hot、word2vec、fastText 为静态方式，即训练好后，每个词表达固定；
    - 单向性：
      - 因为 one-hot、word2vec、fastText 都是 从左向右 学习，导致该方法 不能 同时考虑 两边信息；
  - Elmo 介绍篇
    - 2.1 Elmo 的 特点?
      - 基于特征融合 的 word emb
    - 2.2 Elmo 的 思想是什么?
      - 预训练时，使用语言模型学习一个单词的emb（**多义词无法解决**）；
      - 使用时，单词间具有特定上下文，可根据上下文单词语义调整单词的emb表示（**可解决多义词问题**）
        - 理解：因为预训练过程中，emlo 中 的 lstm 能够学习到每个词 对应的 上下文信息，并保存在网络中，在 fine-turning 时，下游任务 能够对 该 网络进行 fine-turning，使其 学习到新特征；



- Elmo 问题篇
  - 3.1 Elmo 存在的问题是什么?
    - 在做序列编码任务时，使用 LSTM；
    - ELMo 采用双向拼接的融合特征，比Bert一体化融合特征方式弱；

以上内容整理于 幕布文档