Fast-text

特征方式有 tf，tf-idf，word2vec，n-grams，之前我们已经介绍了 tf，tf-idf,这次我们聚焦在 fasttext 上。fastText 方法包含三部分：**模型架构**，**层次 SoftMax** 和 **N-gram 子词特征**。

1. 模型架构

fastText 的架构和 word2vec 中的 CBOW 的架构类似，CBOW 的架构**:**输入的是 $w(t)$的上下文 2d 个词，经过隐藏层后，输出的是 w(t)。

2. 层次 SoftMax

对于有大量类别的数据集，fastText 使用了一个分层分类器（而非扁平式架构）。不同的类别被整合进树形结构中（想象下二叉树而非 list）。在某些文本分类任务中类别很多，计算线性分类器的复杂度高。为了改善运行时间，fastText 模型使用了层次 Softmax 技巧。层次 Softmax 技巧建立在哈弗曼编码的基础上，对标签进行编码，能够极大地缩小模型预测目标的数量。

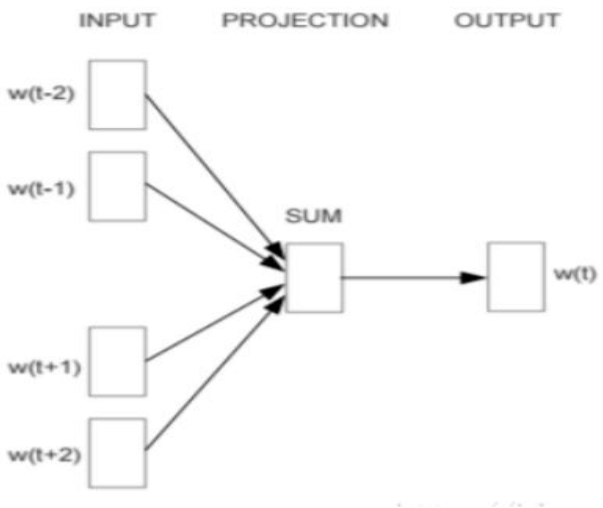
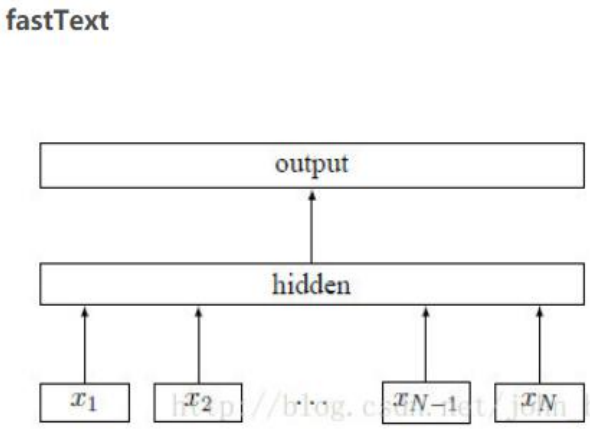
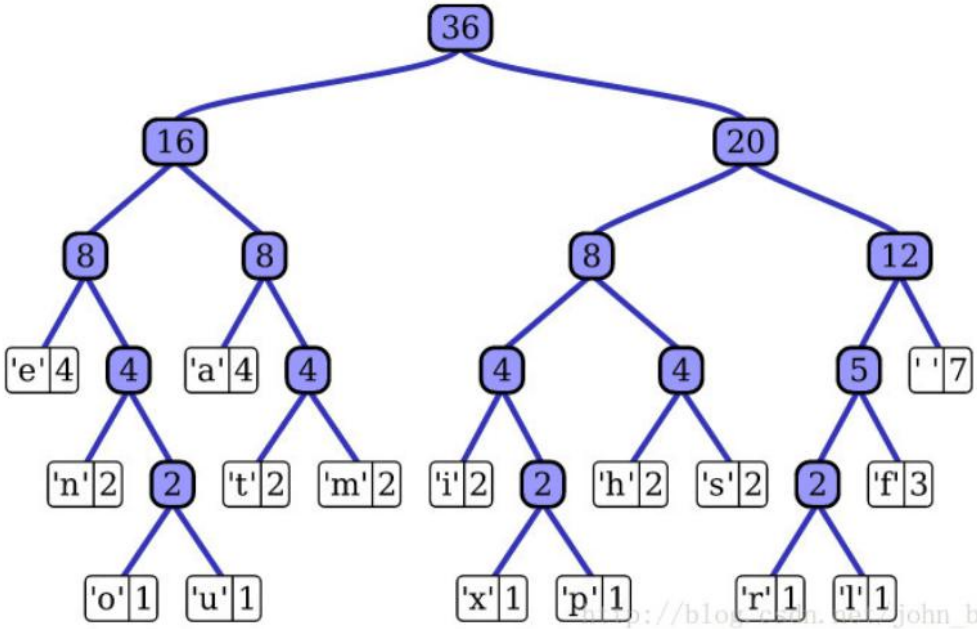


Fig1.cbow

fastText 也利用了类别（class）不均衡这个事实（一些类别出现次数比其他的更多），通过使用 Huffman 算法建立用于表征类别的树形结构。因此，频繁出现类别的树形结构的深度要比不频繁出现类别的树形结构的深度要小，这也使得进一步的计算效率更高。



fastText 模型输入一个词的序列（一段文本或者一句话)，输出这个词序列属于不同类别的概率。序列中的词和词组组成特征向量，特征向量通过线性变换映射到中间层，中间层再映射到标签。fastText 在预测标签时使用了非线性激活函数，但在中间层不使用非线性激活函数。fastText 模型架构和 Word2Vec 中的 CBOW 模型很类似。不同之处在于，fastText 预测标签，而 CBOW 模型预测中间词。



3.N-gram 子词特征

fastText 可以用于文本分类和句子分类。不管是文本分类还是句子分类，我们常用的特征是词袋模型。但词袋模型不能考虑词之间的顺序，因此 fastText 还加入了 N-gram 特征。在 fasttext 中，每个词被看做是 n-gram 字母串包。为了区分前后缀情况，"<"， ">" 符号被加到了词的前后端。除了词的子串外，词本身也被包含进了 n-gram 字母串包。以 where 为例，n=3 的情况下，其子串分别为<wh, whe, her, ere, re>，以及其本身 。

补：n-gram
N-gram 模型是一种语言模型（Language Model，LM），语言模型是一个基于概率的判别模型，它的输入是一句话（单词的顺序序列），输出是这句话的概率，即这些单词的联合概率（joint probability）。

N-gram 中的概率计算
假设我们有一个由 nnn 个词组成的句子 S=(w1,w2,⋯,wn), 如何衡量它的概率呢？让我们假设，每一个单词 wi 都要依赖于从第一个单词 w1 到它之前一个单词 wi−1 的影响:

$$p(S) = p(w_1 w_2 \cdots w_n) = p(w_1)p(w_2|w_1)\cdots p(w_n|w_{n-1}\cdots w_2 w_1)$$

是不是很简单？是的，不过这个衡量方法有两个缺陷:

（1）参数空数过大，概率 p(wn｜wn−1⋯w2w1)的参数有 O(n)个。

（2）数据稀疏严重，词同时出现的情况可能没有，组合阶数高时尤其明显。

为了解决第一个问题，我们引入马尔科夫假设（Markov Assumption）：一个词的出现仅与它之前的若干个词有关。

$$p(w_1 \cdots w_n) = \prod p(w_i|w_{i-1}\cdots w_1) \approx \prod p(w_i|w_{i-1}\cdots w_{i-N+1})$$

那么，如何计算其中的每一项条件概率 p(wn｜wn−1⋯w2w1)呢？答案是**极大似然估计（Maximum Likelihood Estimation，MLE）**，说人话就是数频数:

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

$$p(w_n|w_{n-1}w_{n-2}) = \frac{C(w_{n-2}w_{n-1}w_n)}{C(w_{n-2}w_{n-1})}$$

$$p(w_n|w_{n-1}\cdots w_2 w_1) = \frac{C(w_1 w_2 \cdots w_n)}{C(w_1 w_2 \cdots w_{n-1})}$$

参考资料:

https://blog.csdn.net/songbinxu/article/details/80209197
https://blog.csdn.net/sinat_26917383/article/details/83041424
https://blog.csdn.net/zhouguangfei0717/article/details/81003455
https://blog.csdn.net/feilong_csdn/article/details/88655927
https://www.cnblogs.com/huangyc/p/9768872.html