关键词提取

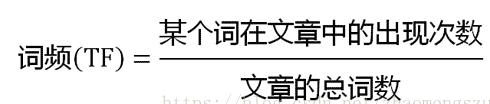
关键词提取与摘要生成是两个较为相似的任务

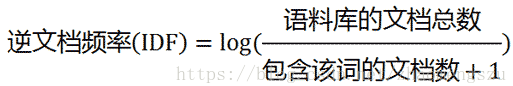
目前主要有以下的处理方法

**1.TF IDF**

参考: https://blog.csdn.net/zhaomengszu/article/details/81452907

计算词的重要性，非监督方法







TF-IDF与一个词在文档中的出现次数成正比，与该词在整个语言中的出现次数成反比(特有信息较为重要，共有信息不重要)

自动提取关键词的算法：计算出文档的每个词的TF-IDF值，然后按降序排列，取排在最前面的几个词。

**2.图论法**

关键词提取算法：textRank与pageRank

参考博客：https://www.cnblogs.com/motohq/p/11887420.html

https://www.cnblogs.com/clover-siyecao/p/5726480.html

https://blog.csdn.net/qq\_41664845/article/details/82869596

3.有监督深度学习方法

Copynet与point generator network都是进行文本摘要任务，但对于关键词提取具有重要参考。

Copynet

CopyNet的优势是 能够比较好的处理 OOV 问题，比如人名、地名等等，因为它可以直接把这部分内容copy到输出中。

原文进行文本摘要任务

参考博客: https://www.sohu.com/a/140364283\_297288

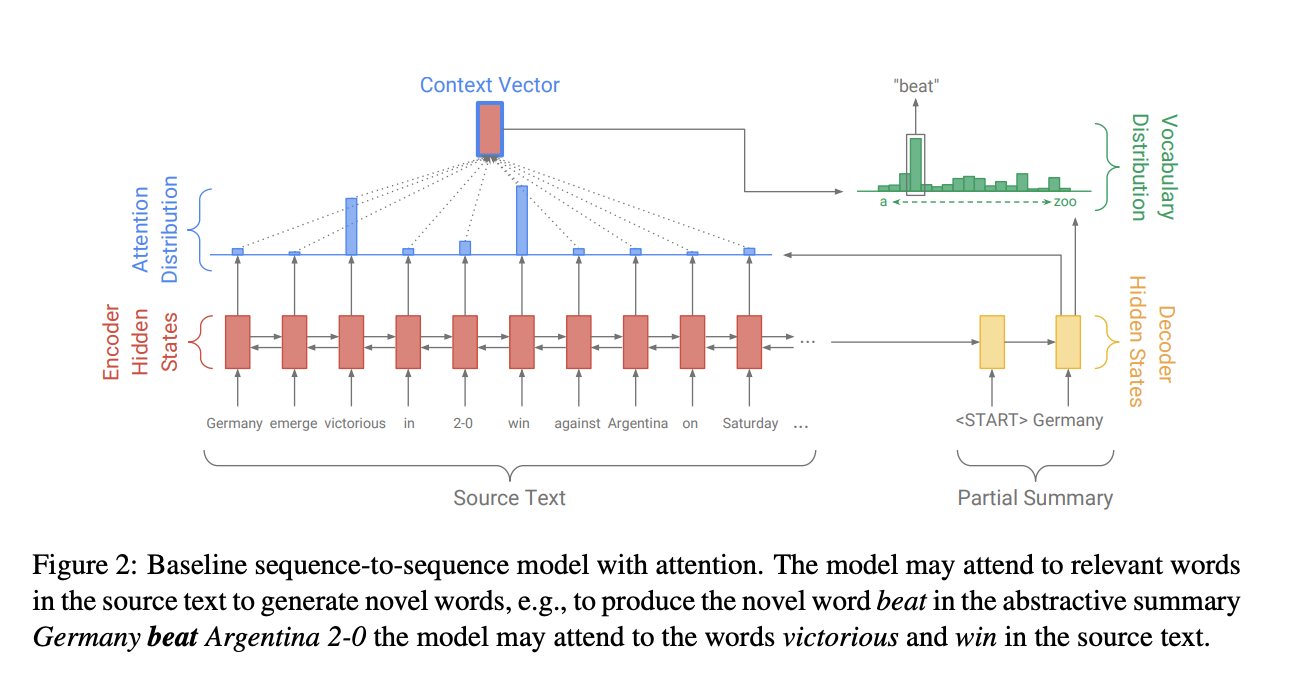
Point generator Network

论文名称：Get To The Point: Summarization with Pointer-Generator Networks

参考博客: https://www.cnblogs.com/zingp/p/11571593.html

Pytorch版本代码(微博新闻数据集): <https://github.com/zingp/pointer-generator-pytorch>

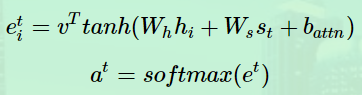
难点：过去S2S模型，难以处理超出字典的词(OOV词)；输出句子中的词语容易重复。

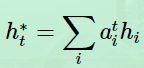


算法框图

注意力生成过程：

首先经过bahdanau注意力



 为注意力权重，为编码器第i步的隐藏状态,ht\*为上下文向量context\_vector(用于生成译码输出)



单词表概率分布

在没有pointer机制中输出词的概率分布则为词典概率分布：





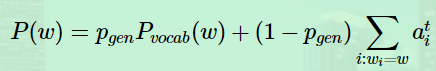
Pointer机制：

判断输出词是生成还是复制

生成概率为：



最终预测概率为:



与普通S2W对比，pointer多了Pgen 生成概率

为原文档中的词，所以(1-Pgen)为复制的概率

Pvocab(w)=0则表示词w不在训练好的词典中。

=0则表示该词不在输入文档中。

Coverage机制：

解决输出词重复，以及未登录词(OOV，超出训练字典)的问题

 coverage\_vector 表示第i个输出词，表示第i个输入词

 有界



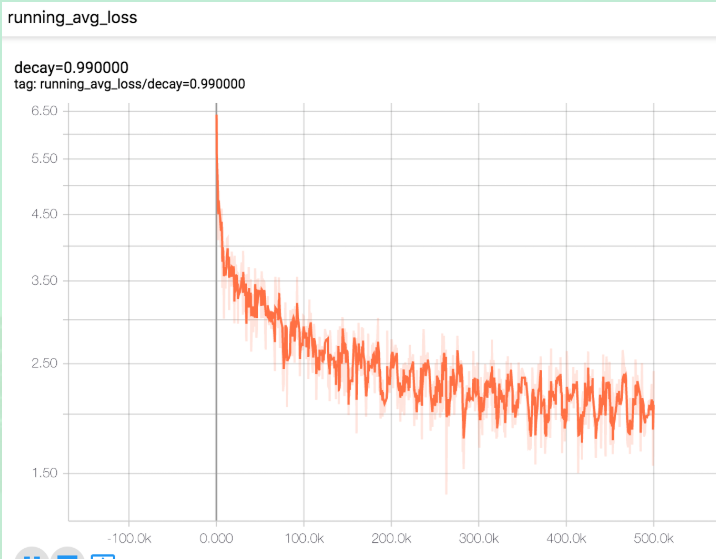
训练方法:

Batch\_size=16

无coverage模式训练500K step，模型收敛，接着coverage模式训练40K

特点：非coverage较容易训练，coverage模式较难训练。

作者的训练过程(微博中文新闻)：

  
2以下比较正常

1. 正则化处理

参考: python正则表达式初识

https://zhuanlan.zhihu.com/p/45088904?utm\_source=wechat\_session&utm\_medium=social&utm\_oi=967002788781363200&utm\_content=first

共10篇

1. 句子embedding

sentence embedding在信息检索，句子匹配，句子分类等任务上均有广泛应用

并且上述任务往往作为下游任务来评测sentence embedding的好坏

词embedding缺乏词序信息的建模，句子emb则补充了这方面的缺点

sentence embedding=句子中所有词的word embedding相加取平均

缺点：每个词重要性相同

论文：A simple but tough-to-beat baseline for sentence embeddings(2016) 提出了加权平均的方法，较容易实现，是目前工程上常用的方法。无监督学习。

该算法分为两步：

1. 对句子中所有的词向量进行加权平均，得到平均向量Vs。(基于TF-IDF方法进行改进，提出 平滑倒词频 (smooth inverse frequency, SIF)的方法 )

词的权重由1/P(w)变为a/(a+P(w))，即频率越低的词在当前句子出现了，说明它在句子中的重要性更大，也就是加权系数更大。

1. 减去Vs在所有句子向量组成的矩阵的第一个主成分(principal component / singular vector)上的投影

核心：如果把一个句子认为是一篇文档并且假设该句中不出现重复的词（TF=1），那么TFIDF将演变成IF，即未平滑的倒词频。但是相较于TFIDF这种经验式公式，论文通过理论证明为SIF提供理论依据。对于第二步，个人的直观理解是移出所有句子的共有信息，因此保留下来的句子向量更能够表示本身并与其它句子向量产生差距。(留下特有的信息)



SIF效果:

在大部分数据集上都比平均词向量或者使用TFIDF加权平均的效果好

在使用PSL(使用有监督任务)作为词向量时甚至能达到最优结果(PSL词向量+WR句向量)

代码：https://github.com/PrincetonML/SIF

在无标签训练数据上设计监督学习任务进行学习

其他工作参考: https://www.cnblogs.com/llhthinker/p/10335164.html