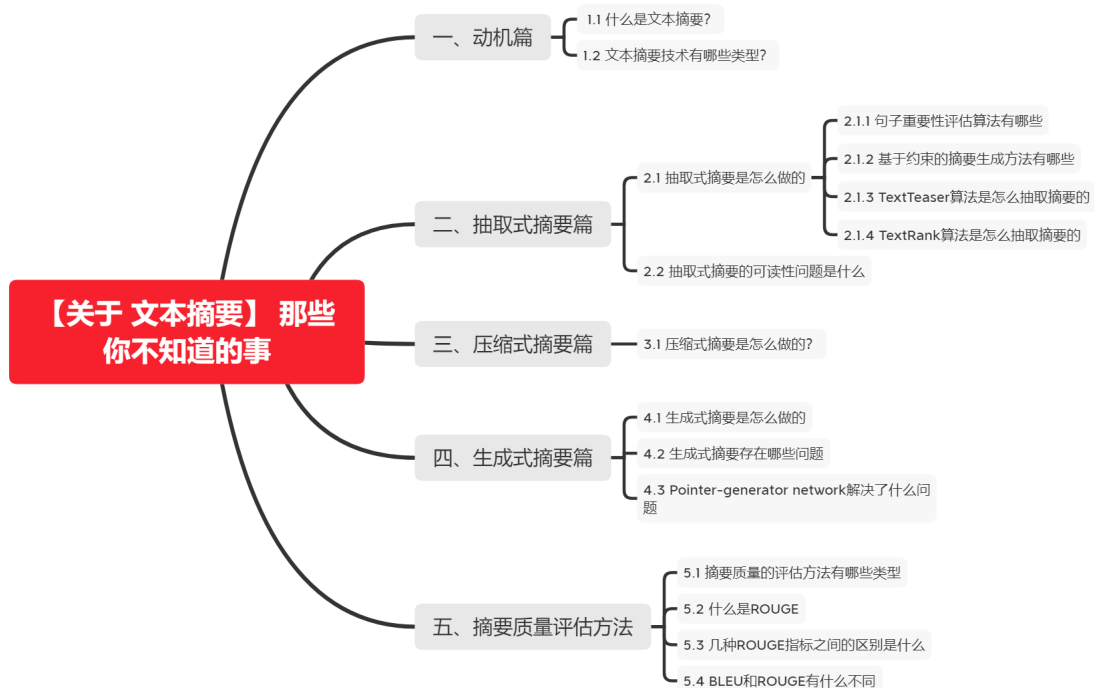


【关于 文本摘要】 那些你不知道的事



一、动机篇

1.1 什么是文本摘要?

文本(自动)摘要利用计算机自动地将文本(或文档集合)转换成简短摘要的一种信息压缩技术。

一般而言, 生成的简短摘要必须满足信息量充分、能够覆盖原文的主要内容、冗余度低和可读性高等要求。

1.2 文本摘要技术有哪些类型?

从不同的角度文本自动摘要技术可以被划分为不同的类型。

按照摘要的功能划分:

- 指示型摘要(indicative)——仅提供输入文档(或文档集)的关键主题, 旨在帮助用户决定是否需要阅读原文, 如标题生成。
- 报道型摘要(informative)——提供输入文档(或文档集)的主要信息, 使用户无需阅读原文。
- 评论型摘要(critical)——不仅提供输入文档(或文档集)的主要信息, 而且需要给出关于原文的关键评论。

根据输入文本的数量划分:

- 单文档摘要(single-document summarization)
- 多文档摘要(multi-document summarization)

根据输入和输出语言的不同划分:

- 单语言摘要(monolingual summarization)——输入和输出都是同一种语言
- 跨语言摘要(cross-lingual summarization)——输入和输出是不同的语言

- 多语言摘要(multi-lingual summarization)——输入是多种语言，输出是其中的某一种语言

根据应用形式的划分：

- 通用型摘要(generic summarization)——总结原文作者的主要观点
- 面向用户查询的摘要(query-based summarization)——提供与用户兴趣密切相关的内容

根据文摘与原文的关系(即文摘获取方法)划分：

- 抽取式摘要(extraction-based summarization)——摘录原文的重要句子形成摘要
- 压缩式摘要(compression-based summarization)——抽取并简化原文的重要句子形成摘要
- 理解式摘要(abstraction-based summarization)——改写或重新组织原文内容形成摘要

按照输出文摘的长度划分：

- 标题式摘要
- 短摘要
- 长摘要

其中抽取式、压缩式、理解式（或称生成式）是常用的分类方法。

二、抽取式摘要篇

2.1 抽取式摘要是怎么做的？

抽取式摘要，即直接从原文中抽取一些句子组成摘要。抽取式的方法基于一个假设，一篇文档的核心思想可以用文档中的某一句或几句话来概括。那么摘要的任务就变成了找到文档中最重要的几句话。

这种摘要方式本质上是是个排序问题，给每个句子打分排序，再考虑冗余性、新颖性、多样性等做一些后处理（比如MMR），抽取高分的句子作为摘要。这个过程涉及到句子重要性评估算法和基于约束的摘要生成算法。

2.1.1 句子重要性评估算法有哪些？

常见的句子重要性评估算法有：

- 基于统计：统计词频、位置、长度、是否包含标题词等信息，计算句子的得分，再选出得分高的句子作为摘要。例如Lead3、TextTeaser等。特点：简单易用，但对词句的使用大多仅停留在表面信息。
- 基于图模型：将句子看作结点，句子之间的相似度看作边，构建图，然后用PageRank算法迭代得到每个句子的得分。例如TextRank, LexRank等
- 基于潜在语义：使用主题模型，挖掘语句的隐藏信息。例如LDA, HMM等
- 基于线路规划：将摘要问题转为线路规划，求全局最优解。
- 基于机器学习：句子分类（摘要句、非摘要句，可视为序列标注问题）、聚类（每个类选择一个距离质心最近的句子）等

2.1.2 基于约束的摘要生成方法有哪些？

排序之后的结果只考虑了相关性并没有考虑新颖性，非常有可能出现排名靠前的几句话表达的都是相似的意思。所以需要引入一个惩罚因子，将新颖性考虑进去。对所有的句子重新打分，如下公式：

$$score(i) = a * score(i) + (1-a) * similarity(i, i-1), i = 2, 3, \dots, N$$

序号*i*表示排序后的顺序，从第二句开始，排第一的句子不需要重新计算，后面的句子必须被和前一局的相似度进行惩罚。

这个算法就是MMR (Maximum Margin Relevance, 最大边缘相关) (注意: 这里的公式不是原始的MMR公式, 而是针对文本摘要任务做了修改)。它是常见的最小化冗余度算法, 主要是面向查询相关的文档自动摘要任务提出。基本思想是在未选句子集合中选择一个与输入查询最相关并且与已选句子最不相似的句子, 迭代该过程, 直至句子数目或单词数目达到上限。

2.1.3 TextTeaser算法是怎么抽取摘要的?

简单来说, 根据词频、位置、长度、是否包含标题词等统计指标, 计算句子的得分, 再选出得分高的句子作为摘要。

具体来说, TextTeaser的统计指标有:

- 1) 句子长度, 长度为某个长度 (比如20) 的句子为最理想的长度, 依照距离这个长度的远近来打分。
- 2) 句子位置, 根据句子在全文中的位置, 给出分数。(比如每段的第一句是核心句的比例大概是70%)。也可以考虑用句子在段落中的位置来评估。
- 3) 文章标题与文章内容的关系, 句子是否包含标题词, 根据句子中包含标题词的多少来打分。
- 4) 句子关键词打分, 文本进行预处理之后, 按照词频统计出排名前10的关键词, 通过比较句子中包含关键词的情况, 以及关键词分布的情况来打分。

综合上述步骤的打分做累加, 然后倒排得到每个句子的重要性得分, 此时要考虑到摘要的可读性, 通俗的做法是按照句子在文章中出现的顺序输出得分最高的n句话作为摘要。

Python代码开源版本: <https://github.com/DataTeaser/textteaser>

2.1.4 TextRank算法是怎么抽取摘要的?

简单来说, 将句子看作结点, 句子之间的相似度看作边, 构建图, 然后用类似PageRank的算法迭代得到每个句子的得分, 最终将得分高的句子输出得到摘要。

详细的步骤如下

1. 预处理: 将输入的文本或文本集的内容分割成句子得 $T=[S\{1\}, S\{2\}, \dots, S\{m\}]$, 构建图 $G=(V, E)$, 其中 V 为句子集, 对句子进行分词、去除停止词, 得 $S\{i\}=[t\{i, 1\}, t\{i, 2\}, \dots, t\{i, n\}]$, 其中 $t\{i, j\} \in S_{\{i\}}$ 是保留后的候选关键词。
2. 句子相似度计算: 构建图 G 中的边集 E , 基于句子间的内容覆盖率, 给定两个句子 $S\{i\}, S\{j\}$, 采用如下公式进行计算:

$$Similarity(S_i, S_j) = \frac{|w_k | w_k \in S_i \cap w_k \in S_j|}{\log(|S_i|) + \log(|S_j|)}$$

S_i : 第 i 个句子。 w_k : 第 k 个单词。 $|S_i|$: 句子 i 中单词数。

简单来说就是, 两个句子单词的交集除以两个句子的长度。也可以用其他计算相似度的方法。

若两个句子之间的相似度大于给定的阈值, 就认为这两个句子语义相关并将它们连接起来, 即边的权值 $w\{i, j\}=Similarity(S\{i\}, S\{j\})$

3. 句子权重计算: 迭代传播权重计算各句子的得分, 计算公式如下:

$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

V_i 为结点。

d 为阻尼系数, 表示从图中某一结点指向其他任意结点的概率。介于0-1之间, 通常设为0.85。

w_{ij} 为 V_i 到 V_j 的边的权重。

$In(V_i)$ 为指向 V_i 的点的集合， $Out(V_j)$ 为 V_j 指出的点的集合。

上面公式与PageRank中的公式相比而言，PageRank中的出度权重每条边的权重都是相等的，在上述公式中使用相似度来进行加权赋值，这样每条出度的边的权重就不一定一样了，而在PageRank中是等权的。

4. 抽取文摘句：将上述得到的句子得分进行倒序排序，抽取重要度最高的T个句子作为候选文摘句。

5. 形成文摘：根据字数或句子数要求，从候选文摘句中抽取句子组成文摘

2.2 抽取式摘要的可读性是什么？

自动文摘质量一直被诟病的问题就是可读性。因为各个句子都是从不同的段落中选择出来的，如果只是生硬地连起来生成摘要的话，很难保证句子之间的衔接和连贯。保证可读性是一件很难的事情。

有一个取巧的方法，就是将排序之后的句子按照原文中的顺序输出，可以在一定程度下保证一点点连贯性。

三、压缩式摘要篇

3.1 压缩式摘要是怎么做的？

压缩式自动摘要对句子进行压缩，保留重要的句子成分，删除无关紧要的成分，使得最终的摘要在固定长度的范围内包含更多的句子，以提升摘要的覆盖度。

核心模块：句子压缩

- 可视为树的精简问题。
- 可视为01序列标注问题。

句子压缩任务可以被定义为一个删词问题：删除句子中不重要的词，形成该句子的一个压缩式表达。常见的方法：

- 基于句法分析树(无监督)：先得到句子对应的短语结构树，再根据规则删除不重要的子树（比如删除介词短语子树、时间短语子树等）
- 基于噪声信道模型(有监督)：给定原始长句子 s ，寻找最佳压缩句子 t ，使得后验概率 $P(t|s)$ 最大（利用贝叶斯准则得到后验概率）。
- 基于决策(有监督)：从结构树改写的角度对句子进行处理，通过一系列“移进-规约-删除”动作实现

压缩式自动摘要方法结合了句子选择和句子压缩两个算法过程，结合方法可以是：（1）先选择后压缩；（2）先压缩后选择；（3）两个过程同时进行。

例如整数线性规划ILP，句子中的每个词都对应一个二值变量表示该词是否保留，每个词都有一个打分（比如tf-idf），目标函数就是最大化句子中的词的打分。最简单的限制比如说至少保留一个词，再比如说当形容词被保留时其修饰的词也要保留（根据parse tree）。

四、生成式摘要篇

4.1 生成式摘要是怎么做的？

生成式摘要，它试图通过理解原文的意思来生成摘要，其实就是模仿人类写摘要的方式。

生成式摘要常见的方法有

- 基于信息融合的生成式摘要：例如基于句法分析树的信息融合技术，利用句法结构树定义概念和事实，计算概念和事实的重要性，度量概念和事实的兼容性，最终组合概念和事实形成摘要句子。

- 基于编码-解码的生成式摘要：在语义向量空间内对文本进行编码，然后通过解码网络逐词生成摘要。

由于深度学习的发展，基于编码-解码的生成式摘要更常见。

4.2 生成式摘要存在哪些问题？

使用seq2seq框架做摘要通常会遇到以下几个问题：

1. OOV问题。源文档语料中的词的数量级通常会很大,但是经常使用的词数量则相对比较固定。因此通常会根据词的频率过滤掉一些词做成词表。这样的做法会导致生成摘要时会遇到UNK的词。
2. 摘要的可读性。通常使用贪心算法或者beamsearch方法来做decoding。这些方法生成的句子有时候会存在不通顺的问题。
3. 摘要的重复性。这个问题出现的频次很高。与2的原因类似，由于一些decoding的方法的自身缺陷，导致模型会在某一段连续timesteps生成重复的词。
4. 长文本摘要生成难度大。对于机器翻译来说，NLG的输入和输出的语素长度大致都在一个量级上，因此NLG在其之上的效果较好。但是对摘要来说，源文本的长度与目标文本的长度通常相差很大，此时就需要encoder很好的将文档的信息总结归纳并传递给decoder，decoder需要完全理解并生成句子。可想而知，这是一个很难的事情。
5. 模型的训练目标与最终的评测指标不太一致。这里牵扯到两个问题，一个是seq2seq的训练模式中，通常会使用teacher-forcing的方式，即在decoder上，将真实target的输入和模型在前一时刻生成的词一起送到下一个时刻的神经元中计算。但是在inference时，是不会有真实target的，因此存在一个gap；另一个问题就是通常模型训练的目标函数都是交叉熵损失函数。但是摘要的评测却不是以交叉熵来判断的，目前一些榜单通常以ROUGE、BLEU等方式评测，虽然这些评测也不是很完美，但是与交叉熵的评测角度均在较大差异。

4.3 Pointer-generator network解决了什么问题？

指针生成网络从两方面针对seq-to-seq模型在生成式文本摘要中的应用做了改进。

第一，使用指针生成器网络可以通过指向从源文本中复制单词(解决OOV的问题)，这有助于准确复制信息，同时保留generator的生成能力。PGN可以看作是抽取式和生成式摘要之间的平衡。

通过一个门来选择产生的单词是来自于词汇表，还是来自输入序列复制。

第二，使用coverage跟踪摘要的内容，不断更新注意力，从而阻止文本不断重复(解决重复性问题)。利用注意力分布追踪目前应该被覆盖的单词，当网络再次注意同一部分的时候予以惩罚。

五、摘要质量评估方法

5.1 摘要质量的评估方法有哪些类型？

1.人工评价方法

请专家对系统的自动摘要结果打分，但是专家之间差异性较大。解决方法之一是金字塔方法(pyramid method)。m位专家撰写参考摘要，然后人工分析每个参考摘要，提取摘要内容单元(summary content unit, SCU)（表示摘要中子句级的重要语义单元）的集合，并为参考摘要中每个SCU赋值，被w个参考摘要提及则赋值为w。然后计算所有SCU在系统摘要中的得分之和，系统摘要得分与理想摘要得分的比值作为质量评价得分。

2.自动评价方法

(1) 内部(intrinsic)评价：分析摘要的质量评价

- 形式度量(form metrics)：侧重于语法、摘要的连贯性和组织结构

- 内容度量(content metrics): 侧重内容和信息, 比如ROUGE (Recall-Oriented Understudy for Gisting Evaluation) 。

(2) 外部(extrinsic)评价: 依据摘要结果对其他应用任务的效果评价

5.2 什么是ROUGE?

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) 是评估自动文本摘要和机器翻译的一组指标。它通过将待审摘要或翻译与一组参考摘要或翻译进行比较计算, 得出相应的分数, 以衡量自动生成的摘要或翻译与参考摘要之间的相似度。

它的基本思想是将待审摘要和参考摘要的n元组共现统计量作为评价依据。然后通过一系列标准进行打分。

ROUGE包括: ROUGH-N、ROUGH-L、ROUGH-W、ROUGH-S和ROUGH-SU几个类型。通俗地讲就是通过一些定量化的指标来描述待审摘要和参考文摘之间的相似性, 维度考虑比较多, 在一定程度上可以很好地评价产生的摘要。

5.3 几种ROUGE指标之间的区别是什么?

ROUGE是将待审摘要和参考摘要的n元组共现统计量作为评价依据。

ROUGE-N = 每个n-gram在参考摘要和系统摘要中同现的最大次数之和 / 参考摘要中每个n-gram出现的次数之和

分母也可以是待审摘要中每个n-gram出现的次数之和, 不过用参考摘要的话更看重召回, 用待审摘要则更看重精确率。

$$Rouge - N = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count(gram_n)}$$

ROUGE-L计算最长公共子序列的匹配率, L是LCS (longest common subsequence) 的首字母。如果两个句子包含的最长公共子序列越长, 说明两个句子越相似。

$$P_{ICS} = \frac{LCS(X, Y)}{n}$$

$$R_{ICS} = \frac{LCS(X, Y)}{m}$$

$$F_{ICS} = \frac{(1 + \beta^2) R_{ICS} P_{ICS}}{R_{ICS} + \beta^2 P_{ICS}}$$

其中LCS(X,Y)是X和Y的最长公共子序列的长度, m,n分别表示参考摘要和待审摘要的长度 (一般就是所含词的个数) 。R和P分别表示召回率和精确率, F即是Rouge-L。一般只考虑召回率, 所以参数 β 会被设置为一个很大的数。

Rouge-W是Rouge-L的改进版, 使用了加权最长公共子序列(Weighted Longest Common Subsequence), 连续最长公共子序列会拥有更大的权重。

ROUGE-S使用了skip-grams。在参考摘要和待审摘要进行匹配时，不要求gram之间必须是连续的，可以跳过几个单词。比如skip-bigram，在产生grams时，允许最多跳过两个词。比如“cat in the hat”的skip-bigrams 就是“cat in, cat the, cat hat, in the, in hat, the hat”。

5.4 BLEU和ROUGE有什么不同？

BLEU 是 2002 年提出的，而 ROUGE 是 2003 年提出的。

BLEU的计算主要基于精确率，ROUGE的计算主要基于召回率。

ROUGE 用作机器翻译评价指标的初衷是这样的：在 SMT（统计机器翻译）时代，机器翻译效果稀烂，需要同时评价翻译的准确度和流畅度；等到 NMT（神经网络机器翻译）出来以后，神经网络脑补能力极强，翻译出的结果都是通顺的，但是有时候容易瞎翻译。

ROUGE的出现很大程度上是为了解决NMT的漏翻问题（低召回率）。所以 ROUGE 只适合评价 NMT，而不适用于 SMT，因为它不管候选译文流不流畅。

BLEU的计算公式：

$$BLEU = BP \times \exp \left(\sum_{n=1}^N W_n \times \log P_n \right)$$
$$BP = \begin{cases} 1 & lc > lr \\ \exp(1 - lr/lc) & lc \leq lr \end{cases}$$

lc = 机器译文的长度
 lr = 最短的参考翻译句子的长度

- BLEU 需要计算译文 1-gram, 2-gram, ..., N-gram 的精确率，一般 N 设置为 4 即可，公式中的 P_n 指 n-gram 的精确率。
- W_n 指 n-gram 的权重，一般设为均匀权重，即对于任意 n 都有 $W_n = 1/N$ 。
- BP 是惩罚因子，如果译文的长度小于最短的参考译文，则 BP 小于 1。
- BLEU 的 1-gram 精确率表示译文忠于原文的程度，而其他 n-gram 表示翻译的流畅程度。

BLEU一般使用改进的方法计算精确率。给定参考译文 S_1, S_2, \dots, S_m ，待审译文 C ，可以计算 C 里面 n 元组的 Precision：

$$P_n = \frac{\sum_{i \in n_gram} \min(h_i(C), \max_{j \in m} h_i(S_j))}{\sum_{i \in n_gram} h_i(C)}$$

i : C 里面的 n_gram
 $h_i(C)$: n_gram i 在 C 里面出现的次数
 $h_i(S_j)$: n_gram i 在第 j 个参考译文出现的次数

参考资料

1. 《文本数据挖掘》宗成庆等人著
2. [textteaser算法学习](#)
3. [TextRank算法](#)
4. [自动文摘评测方法：Rouge-1、Rouge-2、Rouge-L、Rouge-S](#)

5. [文本生成13: 万字长文梳理文本生成评价指标](#)
6. [文本自动摘要任务的“不完全”心得总结](#)
7. [真正理解指针生成网络 Pointer-Generator Networks](#)