# 《数据科学与工程算法基础》实践报告

报告题目：应用多种子模函数最大化算法及其改进进行文本摘要

姓 名： 周辛娜

学 号： 10195501442

完成日期： 2021.12.22

提供中英文摘要

摘要 [中文]：

本次实验采用六种方法实现了文本摘要，分别是课本上原始爬山算法(采用集合覆盖函数为子模函数的爬山算法)、通过lazy改进的课本上的爬山算法、modified-greedy algorithm算法、通过lazy改进的modified-greedy algorithm算法、基于Monotone Submodular Objectives的爬山算法、通过lazy改进的基于Monotone Submodular Objectives的爬山算法。本次实验爬取了正点财经10页的内容，每一页有40篇文章，每一篇文章都围绕着一个主题展开。本次项目采用多种子模函数最大化算法及其改进提取了100篇摘要，每一篇摘要都对应着每页的4篇文章，每一页对应10篇摘要，10页刚好100篇。前面两种方法均对爬取的语料库进行了文档摘要，并比较最后结果的可读性以及算法的运行时间。重点是关注后面四种方法。针对后面四种方法，先选取最优的参数，然后再产生100篇摘要，比较四种方法的覆盖率以及运行效率。

Abstract [英语]：

In this experiment, six methods were used to achieve text summarization. They are the original hill-climbing algorithm in the textbook (which uses the set covering function as a submodular function), the hill-climbing algorithm in the textbook improved by lazy, the Modified-greedy algorithm and the Modified-greedy improved by lazy A mountain climbing algorithm based on Monotone Submodular Objectives and improved by lazy. This experiment took 10 pages of zhengdian Finance, each page has 40 articles, each article around a theme. In this project, 100 abstracts were extracted by using the multi-seed modulus function maximization algorithm and its improvement. Each abstract corresponds to 4 articles on each page, and each page corresponds to 10 abstracts, which is exactly 100 articles on 10 pages. The first two methods summarize the crawling corpus and compare the readability of the final results and the running time of the algorithm. It is important to focus on the last four methods. For the last four methods, the optimal parameters were selected first, and then 100 abstracts were generated to compare the coverage and operation efficiency of the four methods.

1. 项目概述（阐明该项目的科学和应用价值，以及相关工作进展并描述项目的主要内容）

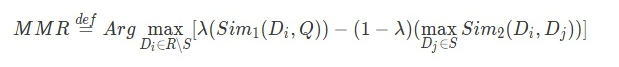
文本摘要可以被抽象为子模函数优化问题，每次选择摘要句的时候，根据子模函数求得句子的权重分数，从而选择权重分数最大的句子放到摘要集合中。那么采用何种子模函数以及如何优化爬山算法以加快算法性能是非常重要的问题，对实际的场景有着深刻的应用价值。

1. 问题描述（问题定义）
2. **抽取式文本摘要问题定义及课本给出的爬山算法(采用集合覆盖函数为子模函数的爬山算法)：**

给定用户定义的n个关键词组成的集合W和m个候选句子的集合S，其中Si表示该句子中包含的关键词集合，抽取式文本摘要问题是要选择k个句子，使得这k个句子能够包含最多数量的关键词。课本给出的爬山算法比较简单，选用的子模函数就是集合覆盖函数。

1. **modified-greedy-algorithm的提出：**

子模函数选用集合覆盖函数仅仅考虑的是覆盖率，但是在实际对生成的摘要质量进行评价时，还要考虑到摘要句子的多样性，那么这时就会使用的MMR算法，希望提取的不同句子描述的是不同的方面。基于MMR的submodular函数：



但这个函数不是一个单调函数，因此爬山算法无法在常数级的量级上逼近最优解，每个迭代并不能保证选择的句子在当前约束下是最优的。故需采用modified-greedy-algorithm算法[1]。后面通过lazy去改进算法的性能。

1. **基于Monotone Submodular Objectives算法的提出：**

基于MMR的submodular函数是不单调的，便希望能找到本身就是单调的submodular函数，这样就能过通过爬山算法，以常数项因子逼近最优解。因此，需要将submodular函数设计成单调函数。故将原本基于MMR的submodular函数中的冗余度函数设计成奖励多样性函数，使得其和相关性函数共同组成一个单调函数[2]。后面同样通过lazy去改进算法的性能。

1. **采用lazy-greedy-algorithm改进以上三种爬山算法：**

如果采用原始的爬山算法，效率非常地下，所以采用lazy进行改进，可以在得到同样的结果下大大提升算法的运行效率[3]。

1. 方法（问题解决步骤和实现细节）

**1）从web中爬取文本数据：**



可以看到，news、news1、news2、news3分别代表4篇文章，它们一起构成一个items，后面会用不同的方法从这合在一起的4篇文章中去生成一篇摘要，一共爬取了10页，每一页有40篇文章，4篇文章生成一篇摘要，那么一共可以生成100篇摘要。

**2）对爬取到的文本进行预处理：**

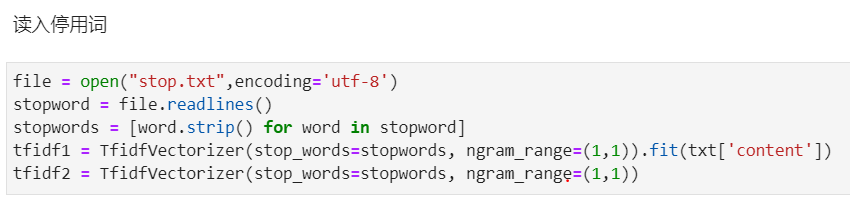
1. 由于文本中有很多如\r，\n，\t等的字符，需要去除



1. 使用jieba库分词



1. 去除停用词



1. 提取文档特征，提取关键词等

通过单词计数向量、TF-IDF方法来提取文档特征，在sklearn里面如下导入：

from sklearn.feature\_extraction.text import TfidfVectorizer

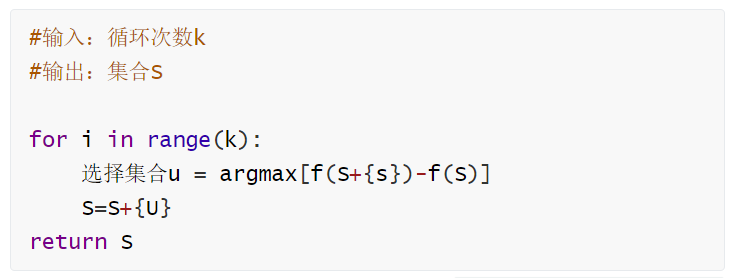
from sklearn.feature\_extraction.text import CountVectorizer

**3）采用集合覆盖函数为子模函数的爬山算法实现：**

1. 爬取语料库，提取关键词，得到关键词集合W

1. 将文档里的所有包含关键词的句子汇总成集合S
2. 在每次迭代中选择使得集合覆盖函数f(A)边际增幅最大的候选元素，通过不停的迭代，从一个候选结果向另一个候选结果移动直到终止条件满足。

伪代码如下：



具体实现：



列表W代表关键词集合，包含了100个关键词。

列表true\_false\_list与关键词集合即列表W的长度相同，用来记录某一个关键词是否已经在被选择的句子中了，初始化均为false。

列表sentence代表候选句子。

列表cover\_list\_number代表某一候选句子包含的关键词的个数。

二维数组cover\_list[i][j]代表第i+1个候选句子的第j+1个词在关键词集合W中的位置，例如cover\_list[0][1]=3代表句子1的第二个关键词位置是关键词集合中的第3个词。

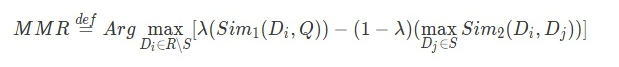
current\_selected代表现在选择第几个句子 。

每次遍历查找使得增加的关键词最多的句子，如果在true\_false\_list中已经有了这个关键词，那么这个句子中包含的关键词个数就要减1，因为是要看新增的关键词个数，为此还要维护一个origin\_cover\_list\_number，选出该句子后，将cover\_list\_number中的对应位置置为0，这样下一次current\_selected就不会选择它了，这样不断迭代直到选出了k个句子。

**4）modified-greedy-algorithm的算法实现：**

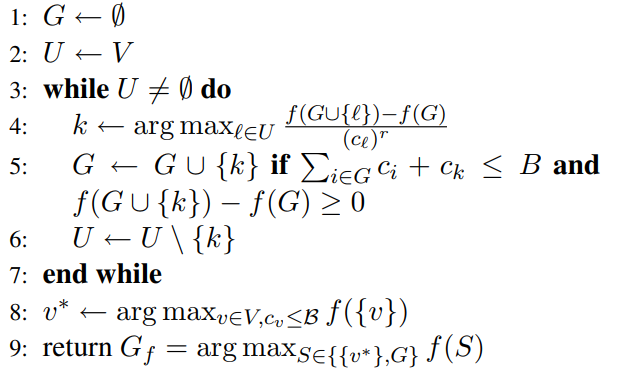
这里的子模函数分为两部分：相关性-冗余性

基于MMR的submodular函数：



第一部分代表的是候选句子和原文的相似度，第二部分代表的是候选句子与当前已有的摘要集合的冗余性，这里相似度采用余弦相似度。

算法伪代码如下：



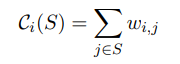
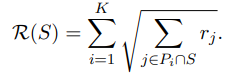
modified-greedy-algorithm在计算候选摘要句边际增幅时，因为每个句子的cost都不一定相同引入scaling factor r进行rescaling。scaling factor能够平衡一些在选择时难以取舍的例子。由于本次实验没有验证集无法通过交叉验证的方式确定最优r，这里就通过运行效率来确定r。

伪代码第8行，跳出最外层循环后，获取边际增幅最大且cost满足约束条件的原始文章中的句子，取该句子对应的submodular函数值与摘要集合的函数值的最大值。这一步能够保证在r=1时得到常数项的逼近因子，克服了由于该子模函数不是一个单调函数而带来的无法收敛的问题。

**5）基于Monotone Submodular Objectives算法实现：**

这里子模函数分为两部分：相关性+冗余性

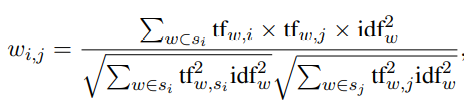


其中，S表示候选摘要集，L(S)度量覆盖率，R(S)用于奖励多样性，λ>=0，通过奖励多样性的方式而非冗余惩罚的方式保证了该目标函数是单调的，从而用爬山算法可以得到最优解。其中，wi,j表示句子i和句子j之间的相似度，Ci(S)表示原文中待选的句子i与候选摘要集S的相似度，αCi(V)是Ci(S)能够取到的最大值，一旦L(S)=αCi(V)，说明即使将句子i添加到S中，覆盖率也不会有明显提升，从而继续挑选V中剩余的句子。

ri表示对句子i的奖励，评估了句子i对于摘要的重要性。

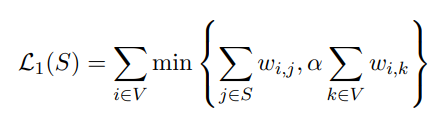
在具体实现中，相似度的度量同论文中采用的相似度相同，为余弦相似度，即：



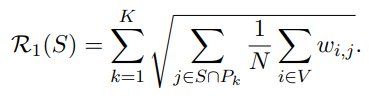
ri为一个候选句子与文档其他句子相似度的矩阵



那么目标函数中的相关性部分就变为了：



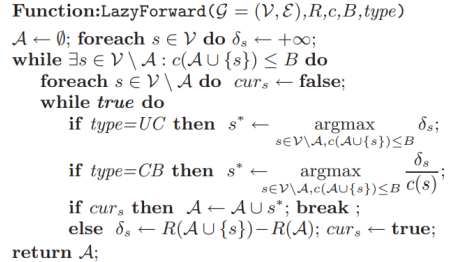
冗余部分变为了：



该算法的整体逻辑同modified-greedy-algorithm，仅仅是最大化的目标函数发生了改变。

**6）lazy greedy algorithm实现**

算法伪代码如下：



原始的爬山算法每次迭代都要计算所有句子的边际增幅，而lazy greedy algorithm每个迭代不一定会计算所有句子的边际增幅。

进行每一次迭代时，都将上一轮迭代得到的句子的权重分数进行排序，选出分数最大值对应的句子，之后，优先更新分数第二大的句子的边际增幅，以此类推。

由于目标函数是单调的，所以分数第二大的句子仍然可能是候选句子中分数最大的句子，从而避免了每次都要计算所有句子的边际增幅，当然最坏情况下，也是要计算所有句子的边际增幅，但这种情况毕竟少数，这样一来，算法的效率就能够大大提高了。

lazy greedy algorithm可以用来加速上述三种方法，所以就另外产生了三种改进方法。

以上是六种方法实现的步骤，具体代码见本文附录。

**7）性能指标**

比较最后结果的可读性、覆盖率以及算法的运行时间。运行时间是利用python中的datetime模块的datetime.now()函数得到当前和结束时间，两者作差便得到运行时间。

1. 实验结果（验证提出方法的有效性和高效性）
2. **首先对四篇文章采用6种不同的方法生成一篇摘要，来比较生成摘要的时间及可读性**
3. 运行时间：

采用集合覆盖函数为子模函数的爬山算法：



经lazy加速后的采用集合覆盖函数为子模函数的爬山算法：

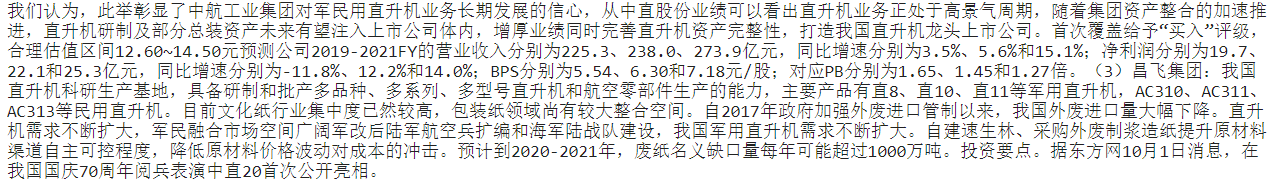


可以看到，经lazy加速后算法效率提高了。

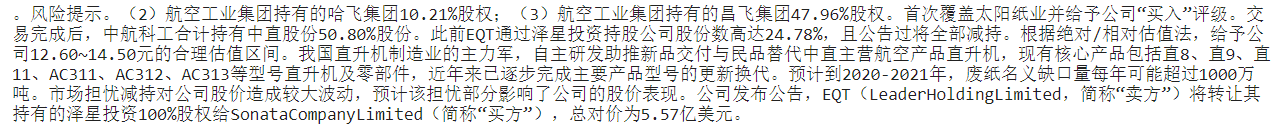
1. 可读性:

由于lazy仅仅是加速了算法，并没有改变最后的结果，所以这里仅仅显示生成的3个摘要。

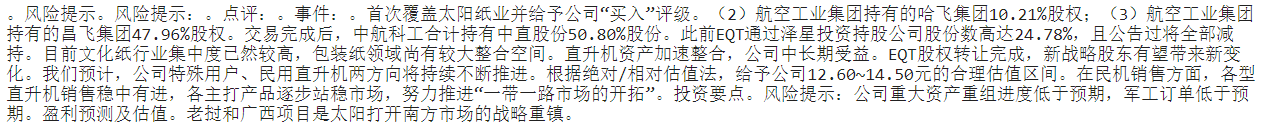
以集合覆盖函数为子模函数的爬山算法：



modified-greedy-algorithm：



基于Monotone Submodular Objectives算法



可以看到，经过lazy加速后的爬山算效率提升了，以集合覆盖函数为子模函数的爬山算法、modified-greedy-algorithm和基于Monotone Submodular Objectives算法的可读性一般。仅仅使用集合的覆盖函数作为子模函数去完成文本摘要事实上还有很多问题没有考虑， 所以接下来主要是比较后面四种方法(modified-greedy algorithm算法、通过lazy改进的modified-greedy algorithm算法、基于Monotone Submodular Objectives的爬山算法、通过lazy改进的基于Monotone Submodular Objectives的爬山算法)。

1. **在不同的超参数取值r下，由运行时间和覆盖率决定最终产生100篇摘要的r值**
2. 运行时间

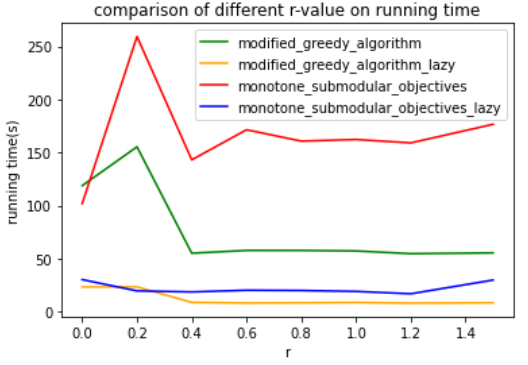


图1

超参数取值r从0变化到1.5过程中，可以清楚看到，modified\_greedy\_algorithm运行时间是要比基于Monotone Submodular Objectives算法要低的，并且经过lazy改进后的两种算法运行时间都比较低，可以说通过lazy使得算法效率改进了若干倍，同时当r取值为0.4-1.4之间时，四种算法的运行效率基本都处于稳定状态。

1. 覆盖率

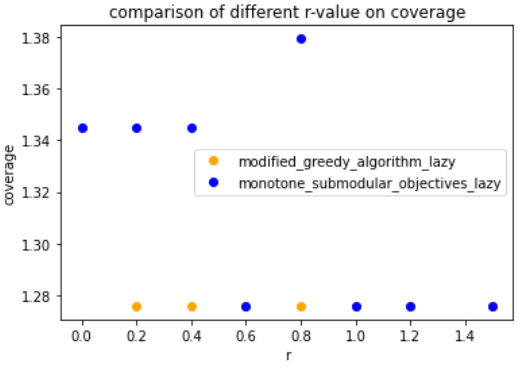


图2

超参数取值r从0变化到1.5过程中，可以看到，基于Monotone Submodular Objectives算法在覆盖率的表现上基本都要比modified-greedy-algorithm好。

综上1) 2)，r=0.8的时候表现整体最优，故确定scaling factor r为0.8。

1. **选定r=0.8后，用后面四种方法对400篇文章生成100篇摘要，进而比较随着文章长度的变化，算法运行时间以及覆盖率的变化**

1）运行时间：

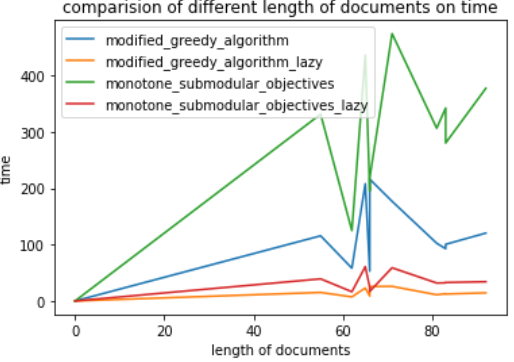


图3

可以看到，随着文章的长度逐渐增加，没有经过lazy改进的两种算法的波动幅度都很大，并且波动曲线也比较相似，都是在length of documents约为60达到极小点，然后又猛增，在length of documents约为65时达到极大点，又下降，尽管经过lazy改进的算法也是在这个区间有所波动，但是都比较平稳，同时不管有没有通过lazy改进，基于Monotone Submodular Objectives算法的运行时间都是要高于modified-greedy-algorithm的。

1. 覆盖率

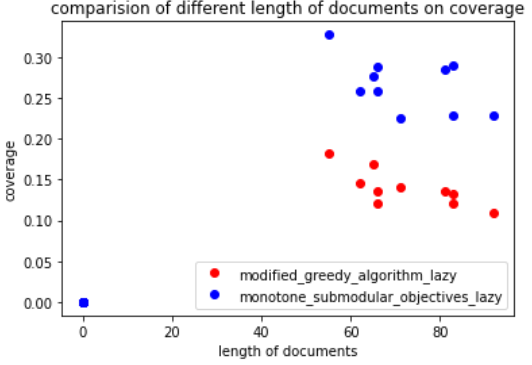


图4

可以看到，随着文章的长度逐渐增加，经过lazy改进的基于Monotone Submodular Objectives算法的覆盖率都是要高于modified-greedy-algorithm的，而且甚至高达2倍。

综合考虑以上2个评价指标，经过lazy改进的基于Monotone Submodular Objectives算法表现最优，不仅有着较高的覆盖率而且运行效率也较高。同时也可以看到lazy改进的强大作用。

1. 结论（对使用的方法可能存在的不足进行分析，以及未来可能的研究方向进行讨论）

在本次实验爬取了正点财经10页的内容，采用多种子模函数最大化算法及其改进提取了100篇摘要，共六种方法实现文本摘要。在算法效率与覆盖率上经过lazy改进的基于Monotone Submodular Objectives算法表现最好。事实上，关于新的子模函数以及优化算法还有很多，那么针对什么样的应用究竟选择何种子模函数和何种优化方法就显得非常有意义了。此外，关于文本摘要问题也可以采用其他方法，如常见的TextRank算法，那么它同子模优化的方法孰优孰劣也是一个较好的研究方向。

[参考文献]

[1] Hui Lin, Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), 2010.

[2] Hui Lin, Jeff Bilmes. A Class of Submodular Functions for Document Summarization. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 510-520, Jun. 2011.

[3] [Jure Leskovec,](javascript:void(0);" \o "Jure Leskovec) [Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, Natalie Glance,](javascript:void(0);" \o "Andreas Krause) Cost-effective Outbreak Detection in Networks. [Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining](https://dl.acm.org/doi/proceedings/10.1145/1281192" \o "KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining). 420-429, August 2007.