Python 程序设计 project – 自然语言处理

2021 年 5 月 19 日

1 引言

Python 作为目前人工智能领域主流语言，在自然语言处理上有着很多应用。本次大作业要求使用 python 语言实现对文本进行自动关键词、关键短语、关键句的提取。这个问题涉及到数据挖掘、文本处理、信息检索等很多计算机前沿领域，但是出乎意料的是，有一个非常简单的经典算法，可以给出令人相当满意的结果。它简单到都不需要高等数学，普通人只用 10 分钟就可以理解，这就是 **TF-IDF** 算法。

2 正则表达式进行文本预处理 (30 分)

(1) 分句操作。对于一篇文档而言往往是没有句子边界的需要自己去 split，这里为了简单起见我们认为 '.', '?', '!' 这三个符号 + 空格 + 一个以大写字母为开头的单词可以作为一个句子的划分。(10 分

Input: This is sentence one. This is sentence two?

Output: [This is sentence one., This is sentence two?]

注意不可简单 split 我们要求保留原句的标点符号。

(2) 异常词。英语中往往标点符号和单词是没有空格隔开的，这样就会导致出现一些奇怪的单词比如明明都是 *end* 但是由于标点符号会出现 *python., python, python!* 因此我们的预处理任务要求分离和单词直接相连的标点符号，经过第一步我们已经可以为文章分句了因此我们的输入输出的示例为：

input: [This is sentence one., This is sentence two?]

Output : [['This' , 'is' ,' sentence' , 'one' , '.'],['This' , 'is' ,' sentence' , 'two' , '?']]

注意有一些符号不能被拆分如作为数字符号不可被分割如 1.5 不能被分成 ['1' , '.' , '5'] U.S.A , 也不能被拆分要求考虑的符号有 ',' '.' '?' '!' 四个标点 (10 分)

(3) 信息匹配. 给定语料中可能带有一些用户个人信息如手机号码和邮箱，我们假定以'1' 或者'+86-' 开头的数字字符串长度大于 9 且小于 12 被认为是手机号，找到之后将其替换为 (phone)，如 +86-1333333333 => (phone) , 18723332333 =>(phone) , 邮箱格式考虑常用邮箱即如 cxan996@qq.com，cxan996@fudan.edu.cn => (email)（10 分）

3 关键词抽取（30 分）

根据助教给定的数据集，对每篇文章抽取 5% 的单词作为关键词，不足 20 个词的文档至少抽取一个。

我们按照 tf-Idf 分数按照每个单词的得分进行排序。首先介绍一下 tf—Idf 算法. 这个算法在网上有非

常多的资料大家可以利用知乎，CSDN 进行学习。

TF-IDF(term frequency–inverse document frequency) TF-IDF 有两层意思，一层是” 词频”（Term Frequency，缩写为 TF），另一层是” 逆文档频率”（Inverse Document Frequency，缩写为 IDF）。假设我们现在有一片长文叫做《python 程序设计》词频高在文章中往往是停用词，“的”，“是”，“了”等，这些在文档中最常见但对结果毫无帮助、需要过滤掉的词，用 TF 可以统计到这些停用词并把它们过滤。当高频词过滤后就只需考虑剩下的有实际意义的词。

但这样又会遇到了另一个问题，我们可能发现” 代码”、” 系统”、” 架构” 这三个词的出现次数一样多。这是不是意味着，作为关键词，它们的重要性是一样的？事实上系统应该在其他文章比较常见，所以在关键词排序上，“代码” 和 “架构” 应该排在 “系统” 前面，这个时候就需要 IDF，IDF 会给常见的词较小的权重，它的大小与一个词的常见程度成反比。

当有 TF(词频) 和 IDF(逆文档频率) 后，将这两个词相乘，就能得到一个词的 TF-IDF 的值。某个词在文章中的 TF-IDF 越大，那么一般而言这个词在这篇文章的重要性会越高，所以通过计算文章中各个词的 TF-IDF，由大到小排序，排在最前面的几个词，就是该文章的关键词。

TF-IDF 算法步骤

1. 对于某给定的单词，第一步，计算词频

$$TF = \frac{\text{该词在文章中出现的次数}}{\text{文章的总词数}} \tag{1}$$

2. 第二步, 计算逆文档频率

$$IDF = log(\frac{\text{语料库的文档总数}}{\text{包含该词的文档数 } +1}) \tag{2}$$

3. 最后一步, 计算该词的 TF-IDF 得分

$$TF - IDF = TF \times IDF \tag{3}$$

注意有些单词出现频率虽然很高如 is are that，但是这些单词不可能为文章的关键词，我们维护一个停用词表，如果某个单词在所有文章中都出现过，那么它不可为关键词。

4 关键句（10 分）

包含最多关键词的句子我们认为是关键句，要求抽取一篇文章的前 TOP 25% 条关键句。少于 4 句话的文章抽取一句。

5 关键短语（20 分）

如果抽取出的关键词在原文中相邻则自动合并为一个关键短语，将其从关键词中移除，计入关键短语，假设我们合并了 3 个单词，那么还需要从后续的备选关键词（按照得分排序）再选 top3 个补上如果构成关键短语还需要重复继续执行以上操作。

如一篇文档的 3 个关键词为 python，programming, NLP. interesting 和 project 的得分排在这三个词后面且 python 和 programming 在原文中相邻，

=> 关键词为 NLP, interesting, project

=> 关键短语：python programming

如果关键词中后续再出现可以合并为短语的重复以上操作，直到关键词中没有可以合并的或者关键短语数量超过所需抽取的关键词数量. 如果关键词中包括多个可以合并的单词，优先合并排在前面的词（以排名最高的词为准，如第一个和第五个可以合并同时第二个和第三个也可以合并，优先合并 1，5）

6 提交要求

数据库文件 `database.txt`, 每行为一个所需提取关键词的 document，稍后助教会发布到 elearning。提交的 python 文件见助教给的 `main.py`. 要求按照实现助教所给的函数，和输出格式。不可使用除 python 自带的第三方库

除了可以运行的代码你还需要提交：1. 详细的实现说明文档需包括核心的代码截图，核心代码的说明 2. 你的详细工作日志包括你遇到了什么问题是如何解决的。文档和日志的编写各 5 分共 10 分。

由于助教测试的时候会使用其他的 databse 因此大家**不要去修改助教文件写的 argparse 和 database 的文件格式, ddl 约四周 6.13 晚 23.59**。