文本复制检测报告单(全文标明引文)

№:ADBD2018R_2018053015312720180530154843440174205963 检测时间:2018-05-30 15:48:43

检测文献: 53141113_王琦_计算机科学与技术（网络与信息安全）_基于强化学习的案件相似度计算方法设计与实现1

作者: 王琦

检测范围: 中国学术期刊网络出版总库
中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库
中国重要会议论文全文数据库
中国重要报纸全文数据库
中国专利全文数据库
图书资源
优先出版文献库
大学生论文联合比对库
互联网资源(包含贴吧等论坛资源)
英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)
港澳台学术文献库
互联网文档资源
CNKI大成编客-原创作品库
个人比对库

时间范围: 1900-01-01至2018-05-30

检测结果

总文字复制比：4.2% 跨语言检测结果：0%

去除引用文献复制比：3.4% 去除本人已发表文献复制比：4.2%

单篇最大文字复制比：0.8%

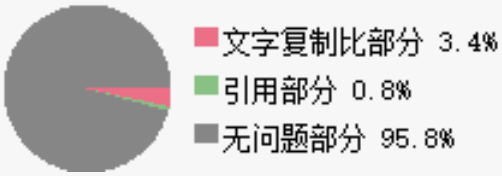
重复字数： [1056] 总段落数： [7]

总字数： [25333] 疑似段落数： [5]

单篇最大重复字数： [190] 前部重合字数： [26]

疑似段落最大重合字数：[492] 后部重合字数： [1030]

疑似段落最小重合字数：[29]



指 标：☐疑似剽窃观点 ☑疑似剽窃文字表述 ☐疑似自我剽窃 ☐疑似整体剽窃 ☐过度引用

表格： 0 公式： 4 疑似文字的图片： 0 脚注与尾注：11

0%（0） 中英文摘要等（总3729字）
4.1%（122） 第1章绪论（总2985字）
9.8%（492） 第2章技术路线（总5022字）
5.4%（300） 第3章基于强化学习的案件相似度计算方法设计（总5534字）
2.3%（113） 第4章基于强化学习的案件相似度计算方法实现（总4854字）
0%（0） 第5章系统测试（总1683字）
1.9%（29） 第6章总结与展望（总1526字）

（注释：无问题部分 文字复制比部分 引用部分）

1. 中英文摘要等 总字数：3729

相似文献列表 文字复制比：0%(0) 疑似剽窃观点：(0)

原文内容 红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

摘要

基于强化学习的案件相似度计算方法设计与实现

近年来随着政府信息公开化、透明化，法院和法官对于各类案件的审理结果越来越受到公众的重视。法官在审理案件时

往往会参考以往类似的案件的审判结果，综合考虑以往的判决结果和社会影响最终做出判决。如果能够为法官提供一个系统，当法官遇到新的案件时，能够为其提供准确的相似案件作为参考，大大减轻法官的工作强度，从繁琐的文书数据中解脱出来，并且对于提高审理案件的准确性和客观公正性有着重要作用。因此，本文提出基于强化学习的案件相似度计算系统，通过结合机器学习和关键词抽取技术，实现了案件相似度计算和相似案件推荐功能。

通过爬虫技术，将互联网上的裁判文书信息爬取到文书库。利用自然语言处理技术，通过现有的中文处理工具，将裁判文书按结构分段，标记，并进行中文分词处理。随后利用计算后的关键词权值建立向量模型。并利用强化学习算法过滤并优化模型内数据。最后通过计算余弦相似度，得到两篇案件文书的相似性。

本文的大量裁判文书主要取自中国裁判文书网。在该网站中用户只可以根据手动输入的关键字进行全文检索，或者手动选择案件类型，案由、法院、裁判年份等限制条件，进行特征检索。这种检索往往是片面的，无法准确概况裁判文书信息。本文提出的基于强化学习的案件相似度计算系统只需将裁判文书上传，利用相似度计算算法，即可快速得出相似度最高的裁判文书。

相比于传统基于词频的文本推荐系统，本系统专注于法院的案件裁判文书这一特定领域。利用领域内的一些特性，经人工筛选构造一个裁判文书领域内的语料库，建立向量模型，并利用强化学习优化模型。之后将学习出来的模型应用于本系统。应用在本系统提取语义层次的关键词也更加准确。进一步利用关键字来进行相似度的计算也取得了非常不错的效果。

关键词：裁判文书，自然语言处理，强化学习，向量模型，相似度计算

Abstract

Design and implementation of case similarity calculation method based on Reinforcement Learning

Recent years, with the openness and transparency of government information, the court's judgments for various cases has attracted more and more attention from the public. When dealing with a case, a judge often refers to the judgments of similar cases in the past and comprehensively considers the decision. If a system can be provided for a judge, when meeting a new case, it can provide him with accurate similar cases as a reference. Greatly alleviated the strength of the judge's work, removed from the tedious document data, and played an important role in improving the accuracy of the trial and the objectivity of the guest. Therefore, this paper proposes a case similarity calculation system based on reinforcement learning. By combining machine learning and keyword extraction, the case similarity calculation and similar case recommendation function are realized.

Through crawler technology, the information of judgment documents on the Internet can be crawled to the library as much as possible. Using Natural Language Processing technology, through the existing mature Chinese processing tools, the judgment documents are segmented, marked, and word are segmented. Then we use the weight of the key words to establish the vector model. And use reinforcement learning algorithm to filter and optimize the data in the model. Finally, the similarity of two case documents is obtained by calculating cosine similarity.

A large number of judicial documents in this article are mainly drawn from the Chinese judicial documents network. In this website, users can only carry out full text retrieval according to the key words of manual input, or choose the type of the case manually, the circumstances of the case, the court, the referee's years and so on, and carry out the feature retrieval. This kind of search is often one-sided and can not accurately describe the information of the judgment. In this paper, the case similarity calculation system based on reinforcement learning only needs to upload the referee documents, and the similarity calculation algorithm can be used to quickly draw the referee documents with the highest similarity.

Compared with the traditional text recommendation system based on word frequency, this system focuses on the specific case of the court's judgment documents. By using some characteristics in the field, the corpus of a judgments' domain is constructed by artificial screening, the vector model is established, and the optimization model is used in the reinforcement learning. After that, the learning model is applied to the system. The keywords used in this system to extract semantic level are more accurate. Furthermore, using keyword to calculate similarity has achieved very good results.

Keywords：Judgments，NLP，Reinforcement Learning，Vector Model， Similarity Calculation

目录

2. 第1章绪论

总字数：2985

相似文献列表　文字复制比：4.1%(122)　疑似剽窃观点：(0)

1	基于协同过滤模型与隐语义模型的推荐系统研究与实现	2.4%（71）
	鲁权(导师：王如龙;莫继红) -《湖南大学博士论文》- 2013-04-15	是否引证：是
2	教育娱乐机器人嵌入式视觉系统的研究与实现	1.6%（49）
	李少军(导师：闵华清;李慧琪) -《华南理工大学博士论文》- 2012-11-01	是否引证：否

第1章绪论

1.1 项目背景

近年来，随着政府信息的公开化、透明化，各级法院对于各类案件的审理结果越来越受到公众的重视。法官在审理案件时往往会参考以往类似案件的裁判文书，观察其审判结果和社会影响，综合考虑做出判决。该系统的提出意在为法官提供一个相似裁判文书推荐系统，在法官遇到新的案件时，能够为其提供准确的相似案件文书作为参考，大大降低了法官的工作强度，从繁琐的文书工作数据中解脱出来。

经调查研究显示，传统的文书推荐系统，主要是基于手动输入关键字的方式进行全文检索。但是，这种关键字检索的方式往往是片面的，局部的，有时甚至法官也无法准确无误的概述案件信息。检索的结果往往不能符合预期的效果，需要再次进行人工筛选和判断，然而人工筛选和判断后也大大存在着无法准确找到相似的案件的可能性。随着机器学习技术和自然语言处理技术的一步步发展，我们可以让机器自动从裁判文书中抓取关键的信息，利用机器学习算法来优化相似度的计算结果，从而可以进一步对比案件的相似度，将相似度最高的案件推荐给法官。

裁判文书的结构往往是较为复杂的，通常这个结构分为多个部分，每个部分的语言描述方式也多种多样，五花八门的。所以如果只是依靠简单的结构解析，是很难准确的提取出能够概括裁判文书大意的信息的。因此，这时就需要运用自然语言处理技术，去分析裁判文书中每一句的分词结构，将关键词进行提取，过滤。本系统主要针对的结构是裁判文书领域的相似度识别，通过构建裁判文书对应的语料库，并且结合行业特性，建立裁判文书对应的模型，就可以很快的实现对于裁判文书其他领域的拓展。

本系统主要的研究内容就是如何将自然语言处理技术和强化学习技术结合并且应用于裁判文书领域的相似度识别上。通过文书切分处理以及中文分词处理后建立向量模型，将向量模型利用强化学习算法优化和过滤，最终通过余弦相似度算法进行计算，根据筛选出的关键词来进行裁判文书相似度的计算。最后将推荐的结果展示给法官，供法官选择出最准确、最适合的裁判参考

1.2 国内外研究现状

大量的裁判文书，无论是纸质版还是电子版，都已经大大超出了法官的阅读和处理能力，如何从这些海量的裁判文书信息中找到最适合的裁判文书已经成为法官和办案人员的迫切需求。随着社会科学技术的发展,各种国家标准、行业标准、企业标准的数据也在迅速增长。面对越来越多的标准数据，用户仍然依赖于手动输入关键词，并进行全文检索来查找他们感兴趣的内容，这是低效，繁琐且不准确的，很难理清裁判文书间之间的相关性。这种传统的文献检索越来越不能适应当前裁判文书信息量庞大的情况。

目前国内最常用的裁判文书网站是中国裁判文书网。该网站采用将案件文书按照其段落结构，分为首部、事实、理由、判决结果和尾部等。并按照此结构分段存入数据库，然后根据手动输入的关键字进行全文检索，或者手动选择案件类型、案由、法院、裁判年份等限制条件，进行条件检索。这种传统的检索系统从文档录入到文档分类，以及新的案件类型的处理等，都需要大量的人工操作和干预，尚未实现智能而精准的查找和推荐功能。

但纵观全互联网，各个领域的推荐系统却有很多。比较著名的亚马逊公司,就是运用和实现***推荐系统的始祖,据市场分析公司Forrester统计数据显示,约有三分之一的用户会根据亚马逊公司的推荐内容购买物品,这是任何广告都不可能做到的成果[]。***而国内的阿里巴巴，京东也都有着各自根据用户点击内容和搜索内容实时推送推荐商品的推荐系统。除此之外,今日头条,微博等也有根据个人爱好和特征而个性化定制的推荐功能。

在自然语言处理方面，由于中文文本词语间没有英文那样简洁明了的空格作为分割，无法从结构层面就解决分词问题，所以准确的中文分词就变得越发重要，是目前各方研究的重点。国内外已有许多自然语言处理工具。比如哈尔滨工业大学社会计算与信息检索中心开发的LTP语言技术平台，提供了适合各种语言平台的API接口，并且有配套的可视化工具给用户以更多选择。还有中科院的NLPIR（又名ICTCLAS2013)。该系统由张华平博士历时十余年打造，支持多种编码形式，实现了包括微博分词、新词发现和关键词提取等一系列功能，也是一款较为成熟的中文分词系统。同时还有支持Python接口的结巴分词，可以选择分词粒度，并可以识别繁体中文，简单易用且准确度较高。新浪云，支持Rest Api调用，虽然分词粒度较大，但在各个领域的分词处理上都取得了稳定的准确率。还有许多譬如搜狗分词、SCWS、腾讯文智等。

目前许多公司和产品将自然语言处理技术应用于语义的识别。例如猿题库利用自然语言处理技术对试题题目进行解析，按照题目类型归档，根据用户需求给出推荐。聘宝则将自然语言处理技术扩展到招聘领域，将NLP应用于招聘文书的识别，通过分析招聘文书的语义，提取关键词，并据此给应聘者推荐相关职位。

1.3 本项目研究内容

本项目所研究的基于强化学习的案件相似度计算系统，是为广大法官设计的法院裁判案件文书推荐系统。该系统可以自动分析处理法官上传的文书，自动提取案件的案由、案情、裁决等关键信息，并经过相似度计算，为法官推送相似度最高的案件。

本项目采用B/S架构，使用python的Django框架，并采用MTV的框架模式。该框架内嵌了大量相关组件，可以自动处理Web开发的许多繁琐事务，可以使开发人员专注于编写应用程序而不需要关注繁琐的配置。系统分为线上和线下两部分。线下利用大量现有的裁判文书建立语料库，训练模型，并通过强化学习算法优化向量模型；线上接收用户上传的文书文件，抽取关键信息，利用训练好的模型，计算案件相似度，并将相似度最高的案件反馈给用户。

系统实现的难点集中在线下部分。首先是裁判文书的预处理工作，如何将复杂的裁判文书根据内容切分为结构化的数据，并根据需求选取必要的数据内容进行中文分词处理。预处理后裁判文书交由模型训练模块，通过TFIDF算法建立初步的向量模型，随后利用强化学习算法优化向量模型，过滤冗余数据。最终将训练好的模型序列化存储。

本系统的线上部分主要负责与用户的交互任务，在接收用户上传的裁判文书文件后，同样需要调用裁判文书的预处理模块，将裁判文书关键信息进行提取。利用训练好的模型，将该裁判文书与文书库中文书一一进行相似度对比，并根据排序，将相似度最高的案件推荐给用户。

1.4 论文的结构及安排

本文的主要结构安排如下:

第一章绪论部分。介绍了项目研究的背景以及国内外的研究现状。

第二章技术路线。主要主要介绍了项目涉及到的相关技术，包括自然语言处理技术、强化学习技术、TFIDF算法和Python的Django框架。

第三章基于强化学习的案件相似度计算方法设计。是主要是根据已有的相关技术，分析了系统需求，提出了基于强化学习的案件相似度计算系统的实现思路。

第四章基于强化学习的案件相似度计算方法实现。该部分根据基于强化学习的案件相似度计算系统的设计方案，介绍了裁判文书抓取模块、模型训练模块、相似度计算模块和Web工程模块的具体实现。

第五章系统测试。该部分对基于强化学习的案件相似度计算系统进行了性能测试，对比了本系统和传统系统的性能。

第六章总结与展望。进行工作总结，对全文的研究加以概括，并作出展望。

指　标
疑似剽窃文字表述

1. 1.4 论文的结构及安排
   本文的主要结构安排如下:
   第一章绪论部分。介绍了项目研究的背景

脚注和尾注

1. [] 鲁权. 基于协同过滤模型与隐语义模型的推荐系统研究与实现[D]. 湖南大学, 2013..

3. 第2章技术路线

总字数：5022

相似文献列表　文字复制比：9.8%(492)　疑似剽窃观点：(0)

1	11303070241-孙乾-迷宫机器人路径学习仿真系统设计与实现 孙乾 - 《大学生论文联合比对库》- 2017-05-24	2.6%（132） 是否引证：否
2	强化学习算法研究 刘忠;李海红;刘全; - 《计算机工程与设计》- 2008-11-28	2.4%（119） 是否引证：是
3	面向农村医疗的信息抽取方法的研究与实现 耿胜男(导师：吴旭) - 《北京邮电大学博士论文》- 2013-12-31	1.4%（70） 是否引证：否
4	基于Django的在线宠物视频网站 李玉乐 - 《大学生论文联合比对库》- 2017-05-05	1.3%（67） 是否引证：否
5	JVM之上的Python Web框架 - elimago的专栏 - 博客频道 - CSDN.NET - 《网络（http://blog.csdn.net）》- 2012	1.2%（59） 是否引证：否
6	基于强化学习的垂直搜索引擎网络爬虫的研究与实现 刘忠(导师：刘全) - 《苏州大学硕士论文》- 2008-04-01	0.9%（46） 是否引证：否
7	使用多分类器进行Deep Web数据源的分类和判定 李志涛(导师：刘全) - 《苏州大学硕士论文》- 2009-05-01	0.9%（46） 是否引证：否
8	自然语言处理技术在中高职课程衔接中的应用 申玫;徐宁;赵晓玲; - 《职业教育研究》- 2015-11-08	0.6%（31） 是否引证：否
9	J2EE集成开发框架及其应用 李晓飞(导师：谢旭升) - 《江西师范大学博士论文》- 2010-06-01	0.6%（31） 是否引证：否
10	MapReduce原理及其在自然语言处理中的应用研究 亢丽芸;王效岳;白如江; - 《情报科学》- 2014-05-05	0.6%（29） 是否引证：否
11	基于Python自然语言处理的文本分类研究 韦文娟;韩家新;夏海洋; - 《福建电脑》- 2016-07-25	0.6%（29） 是否引证：否
12	汉语自动分词中排除歧义字段算法的研究 刘禹孜(导师：何中市) - 《重庆大学硕士论文》- 2005-03-20	0.6%（29） 是否引证：否

第2章技术路线

2.1 自然语言处理

自然语言纷繁复杂，不同的语言有着各自的语言特点。如何用自然语言与计算机进行沟通，也就成了计算机科学领域与人工智能领域中的一个重要方向[]。自然语言处理有着独特的魅力，是融合了语言学、计算机科学、数学等于一体的科学。

自然语言处理的研究可以理解为如何实现人与计算机间的自然语言交流。中文文本词语间没有英文那样简洁明了的空格作为分割，无法从结构层面就解决分词问题。所以准确的中文分词就变得越发重要，成了研究的重点。目前中文分词技术常采用的方法有两种。一是基于字符串匹配的中文分词处理方法，这种方法需要依赖于文本词典。词典通过人工建立，包含了文书库中所有的词语。利用字符串匹配，当匹配到字典中对应项，即标记为一个词语。该方法依赖于词典，缺陷却也在于词典，不同的语料库不同的词典有着不同的分词结果。。二是基于统计的分词方法。该方法简便易行，也最为有效，是目前最常用的分词方法。这种方法通过统计相邻的字在上下文中同时出现的频率，将多次同时出现且连续的多个字作为一个词语。

目前国内外学术界对于自然语言处理的研究已经取得了很大进展，出现了许多自然语言处理工具。

其中包括LTP语言技术平台。该平台是一套开放的中文自然语言处理系统。它是由哈尔滨理工大学社会计算与信息检索中心开发的，历时十年时间。LTP的语言处理结果可以采用XML、PLAIN、JSON和CONLL等多种格式展示。在此基础上，该平台提出了汉语加工的五个核心技术，即词汇、句法和语义。LTP提供了适合各种语言平台的API接口。并且有配套的可视化工具给用户以更多选择。只需在LTP平台注册账号，获取API_KEY，即可通过REST方式调用API获取处理后的结果，简单易用。目前已有多家知名企业和研究机构在使用免费的学术版和付费商用版LTP系统。

同样出色的还有中科院的NLPIR又名（ICTCLAS2013)。该由张华平博士历时十余年打造。主要功能包括中文分词、词性标注、命名实体识别、用户字典功能等。[]支持多种编码形式。最新的功能包括微博分词、新词发现和关键词提取[]。前后内核升级了十余次，也是一款较为成熟的中文分词系统。

还需要介绍的是一个分词效果较好，使用更加方便的Python模块：结巴（JIEBA）。结巴汉语分词采用Trie树结构实现有效的字图扫描，生成一个有向无环图。该图由句子中所有可能的汉字构词情况组成，并采用动态规划方法求出最大概率路径，找出基于词频的最大切分组合。对于未记录的单词，采用基于汉字构词能力的HMM模型，并采用Viterbi算法。[]目前结巴分词针对不同的应用场景推出了不同的分词模式。在精确模式下，分词的准确度最高，该模式也最为常用。在要求高响应速度的环境下，全模式可以满足你的需求。同时，还有基于精确模式并对应搜引擎的搜索引擎模式。

2.2 强化学习

强化学习又称为增强学习（Reinforcement Learning）是机器学习的一个重要分支，是计算机科学与工程计算智能共同体发展起来的一种辅助学习方法。它试图解决决策优化的问题,是AlphaGo等人工智能的核心技术。近年来，随着高性能计算、大数据和深度学习技术的突飞猛进，强化学习算法及其应用也得到更为广泛的关注和更加快速的发展。强化学习可以通过人工智能与环境的交互作用和修改控制策略来解决人工智能领域中的优化问题。强化学习的目的是在与环境交互的过程中找到最佳的策略，最佳的策略也是最大化的奖励。强化学习的灵感来自于自然学习机制和基于环境的激励和惩罚的动物。在控制领域，采用强化学习的方法可以得到自适应和最优的结果。强化学习的学习内容包括如何获得最佳策略，如何将环境映射到动作以获得最大的响应奖励信号。每个动作都与后面的动作有关。

强化学习最经典的算法包括Q-Learning算法，Policy Gradient算法，Sarsa算法，Deep Q Network算法等。

2.2.1 Q-Learning

*Q-Learning算法最早于1989提出。它采用Q(s,*a)值来作为估计函数。通过不断重复决策过程，不断更新Q(s,a)的值。[]它的基本形式如下：

(5)

式中：在状态s下采用动作a所得到的最优奖赏值的和，而在前面我们也定义为在状态s下的最优值函数，故

(6)

*即我们所采用的最优策略*是

(7)

Q-Learning的迭代公式如(8)

(8)

算法会维护一个Q表，运行伊始，每个Q(s,a)值都被置为0。然后在大概率情况下选择最大的 Q(s,a)值，小概率的情况下会随机选择一个Q(s,a)值。举个例子：假设在 s1阶段采取了a2动作，并到达s2，这时我们开始更新用于决策的Q表，接下来并没有在实际中采取任何行为, 想象自己在s2上的两种行为哪一个的Q值大，假设Q(s2, a2)的值比Q(s2, a1)大。所以将Q(s2, a2)乘上一个衰减值并加上到达s2时所获取的奖励R，因为会获取实实在在的奖励R，我们将这个作为我现实中Q(s1, a2)的值根据估计与现实的差距, 将这个差距乘以一个学习效率累加上老的Q(s, a)的值，并将其更新。如此循环迭代，直至学习过程结束。

2.2.1 Sarsa

Sarsa算法的决策部分和 Q-Learning完全一样。同样使用Q表进行行为决策。 在Q表中挑选值较大的动作值施加在环境中来换取奖惩。但是不同的地方在于 Sarsa的更新方式是不一样的。Q-Learning的行动策略采用Epsilon-Greedy策略，而目标策略为贪心策略。而Sarsa则为同策略。我们称Sarsa算法为on-policy，在线学习，而Q-Learnig也叫作 Off-policy，离线学习。

(9)

公式（9）为Sarsa的迭代公式。Q-Learning算法在状态s只是估计了接下来的动作值，却不一定采取这个行动。而Sarsa算法则说到做到，在状态s估算的动作也是接下来要采取的动作。所以我们会稍稍改动，去掉maxQ，取而代之的是在状态s上我们实际选取的动作action的Q值。最后，求出现实和估计的差距并更新Q表里的Q(s, a)。

2.2.3 Deep Q Network

接下来介绍的是Deep Q Network，简称DQN，是一种融合了神经网络和 Q-Learning 的强化学习方法。

在普通的Q-Learning中，处理的问题往往是离散的，维度较低的。我们利用Q表储存动作对。而对于线性连续的状态，Q-Learning算法则不再适用。这时，结合了深度神经网的DQN算法则最适合自动提取负责特征，应对高纬度的复杂问题。Deep Q Network利用一个神经网络代替了Q表，只需将状态参数输入神经网，利用全连接层的网络，即可输出动作的Q值。

DQN对Q-Learning的修改主要体现在以下三个方面。

（1）DQN利用深度卷积神经网络逼近值函数。

（2）DQN利用了经验回放训练强化学习的学习过程。

（3）DQN独立设置了目标网络来单独处理时间差分算法中的TD偏差。

DQN是一个基于Q-Learning的优秀算法，通过经验池解决了相关性和非静态分布问题，并利用TargetNet解决了稳定性问题。该算法具有很强的通用性，可应用于多种不同场合。但同时也存在一定缺点，例如存在着无法应用于连续的动作控制，CNN不一定收敛，需要调整参数等问题。

2.2.4 Policy Gradient

Policy gradient 是RL中另外一个大家族 。Policy Gradients 直接输出动作的最大好处就是，它能在一个连续区间内挑选动作。而基于值的强化学习算法， 比如Q-learning，则无法应对在无穷多的动作中计算价值的情况。Policy Gradient不像Value-based方法(Q-Learning, Sarsa)，但它也要接受环境信息，不同的是它要输出不是action的value，而是具体的那一个action，这样policy gradient就跳过了value这个阶段。Policy gradient最大的一个优势是：策略函数输出一个确定的action,这个action可以是高维连续的，之前我们说到的 value-based方法输出的都是不连续的值，然后再选择值最大的 action。而 policy gradient 可以在一个连续分布上选取 action。

图2-1 policy gradient算法伪代码

如图2-1所示为Policy Gradient的循环逻辑。其中中的表示在状态 s 对所选动作 a 的吃惊度，如果概率越小，反向的(即 -log(P))反而越大。如果在很小的情况下，拿到了一个大的 Reward,也就是大的 V，那就更大,表示更吃惊,那么就需要对这次的参数进行一个大幅修改。这就是的物理意义。

2.3 TFIDF算法

TF-IDF（term frequency–inverse document frequency）是一种用于资讯检索与资讯探勘的常用加权技术。TF-IDF是一种基于词频的统计方法，可以用来计算一份文章中某个特定词语对于整个文章的重要程度。需要注意的是该方法需要语料库的支持。一个词语在文章中的重要程度随著它在文章中出现的次数增加而增加。但与此同时，在语料库其他文章中，该词语则较少出现，成反比关系。[]在文献关键词提取中，TF-IDF算法经常被使用。

因此TF-IDF计算方法公式如下:

(10)

之所以采用词频而非词数作为权值，是因为同一个词语在较长的文件中往往有着更高的出现次数。为了避免文件长度对于算法结果的影响，TFIDF算法采用词频作为权值参与计算。

(11)

逆文档频率即语料库文档总数与包含该词的文档数+1的商的对数，是词语在语料库中出现频率的反映。包含该词的文档数+1是为了防止出现词语不在语料库中，除数为零的情况出现。

(12)

TFIDF算法同样存在一定不足。在实际应用当中，在同一类文档中，某些特征词往往会频繁出现。这种词语往往能够概括该类文档的特征，即同时在特定文章和语料库中保持着较高的出现频率。这样的词语本应具有较高的权重，但在TFIDF算法中，反而会因其在语料库中较高的出现频率而获得较低权值。

2.4 Python的Django框架

Django是一个高级别的Python Web框架，它鼓励快速开发和干净、实用的设计。Django框架可以自动处理Web开发的许多麻烦，因此你可以专注于编写应用程序而不需要关注繁琐的配置。而且它是免费的和开源的。

Django包含以下特点：

1.Ridiculously fast

Django的设计是为了帮助开发人员尽可能快地从概念到完成应用程序。

2.Fully loaded

Django包括几十个可以用来处理常见Web开发任务的额外插件。Django负责用户身份验证、内容管理、站点地图、RSS订阅以及更多的任务，即时可用。

3.Reassuringly secure

Django十分重视系统安全。许多常见的安全错误，如SQL注入、跨站点脚本、跨站点请求伪造和点击劫持等都可以被主动规避，为开发人员减轻了负担。它的用户认证系统提供了一种安全的方式来管理用户帐户和密码。

4.Exceedingly scalable

世界上最繁忙的一些网站使用Django，来快速灵活地满足最重度的数据交互需求。

5.Incredibly versatile

各类公司、组织和政府利用Django构建了各种各样的东西——从内容管理系统到社交网络到科学计算平台等等。

2.5 本章小结

在本章中，主要介绍了本系统涉及到的相关技术。其中自然语言处理技术是项目的基础。在此基础上，利用TFIDF算法进行初步的相似度计算，并利用强化学习优化计算结果。在介绍了基于词频的TFIDF算法相关知识后，本章还重点介绍了强化学习算法，解释了Q-Learning、DQN和Policy Gradient算法的原理。本章最后介绍了Python的Django框架，用于项目的具体实现。

指　标
疑似剽窃文字表述

1. Django框架
   Django是一个高级别的Python Web框架，它鼓励快速开发和干净、实用的设计。Django

脚注和尾注

1. [] 郑逢斌. 关于计算机理解自然查询语言的研究[D]. 西南交通大学, 2004..
2. [] 杨丽萍. 基于中文分词的图文自动匹配方法研究[D]. 福建师范大学, 2009..
3. [] 刘俊. 关键词抽取方法研究及应用[D]. 重庆大学, 2013..
4. [] 杨望, 董国伟, 龚俭,等. 基于机器学习的网页黑链检测算法[C] 信息安全漏洞分析与风险评估大会. 2014..
5. [] 刘忠, 李海红, 刘全. 强化学习算法研究[J]. 计算机工程与设计, 2008, 29(22):5805-5809..
6. [] TF-IDF模型及其算法[EB/OL] . https://blog.csdn.net/Snail_DM/article/details/54981985, 2017..

4. 第3章基于强化学习的案件相似度计算方法设计　总字数：5534

相似文献列表　文字复制比：5.4%(300)　疑似剽窃观点：(0)

1	漏洞检测与控制措施建议集成平台设计与实现 王鑫 - 《大学生论文联合比对库》- 2015-06-05	3.4%（190） 是否引证：否
2	网站漏洞自动测试系统设计与实现 洪榕森 - 《大学生论文联合比对库》- 2016-05-26	2.5%（138） 是否引证：否
3	基于Java语言网页爬虫设计与实现 朱国栋 - 《大学生论文联合比对库》- 2016-03-31	2.5%（138） 是否引证：否
4	王兆海_面向二手房信息数据挖掘的网络爬虫系统设计和实现 王兆海 - 《大学生论文联合比对库》- 2017-05-31	2.5%（138） 是否引证：否
5	爬虫技术浅析 - yishouwangnian的博客 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	2.5%（138） 是否引证：否
6	网络爬虫的研究与应用 劳景堂 - 《大学生论文联合比对库》- 2016-04-30	2.5%（138） 是否引证：否
7	网络爬虫的研究与应用 劳景堂 - 《大学生论文联合比对库》- 2016-05-06	2.5%（138） 是否引证：否
8	2012314211-劳景堂-网络爬虫的研究与应用 劳景堂 - 《大学生论文联合比对库》- 2016-05-10	2.5%（138） 是否引证：否
9	网站漏洞自动测试系统设计与实现 洪榕森 - 《大学生论文联合比对库》- 2016-06-01	1.6%（90） 是否引证：否
10	网站漏洞自动测试系统 洪榕森 - 《大学生论文联合比对库》- 2016-06-08	1.6%（90） 是否引证：否
11	面向动态网页的定向信息提取模型的设计与实现 盛洁(导师：宫继兵) - 《燕山大学博士论文》- 2016-05-01	1.5%（82） 是否引证：否
12	余弦距离算法在固定资产管理系统中文本相似度查询的应用 朱云峰; - 《无锡商业职业技术学院学报》- 2013-12-25	0.7%（40） 是否引证：否

第3章基于强化学习的案件相似度计算方法设计

3.1 系统需求分析

案件裁判文书是审判活动的真实记录，是审判工作和审判监督的重要依据。卷宗记录着每个案件的涉及人员信息，案由，案情，各方证据，审判结果，复审结果等。卷宗内容纷繁复杂，阅读起来十分费时费力。随着政府信息公开化，透明化，法院对于各类案件的审理结果越来越受到公众的重视。法官在审理案件时往往会参考以往类似案件的审判结果，综合考虑做出判决。

目前已有的可供参考的裁判文书系统是中国裁判文书网。该网站采用的将案件文书按照段落结构，分为首部、事实、理由、判决结果和尾部等。根据手动输入的关键字进行全文检索，或者手动选择案件类型，案由、法院、裁判年份等限制条件，进行特征检索。这种检索系统从文档录入到文档分类，以及新的案例类型的整理等，都需要大量人工操作干预，尚未实现智能化。许多时候用户无法准确描述关键词或案件特征时，该系统则无法准确为用户提供检索和推荐结果。

因此，针对法官面对大量案件卷宗，工作量巨大，已有系统又无法完全满足需要的问题，本文提出了基于强化学习的案件相似度计算方法。该系统的主要需求就是，当法官遇到新的案件时，系统主动为其推荐相似度最高的其他裁判文书，为法官的审判提供参考。

3.1.1 功能性需求

如图3-1所示，该系统的功能性需求主要有三个：案件文书的上传，案件文书查看和相似文书推荐。

案件文书的上传工作，其作用是在法官在处理新的案件时，将新的案件文书文档上传到文书相似度计算系统，同时添加到语料库中，以便后续对裁判文书的处理。

在案件文书文件上传之后，法官可以实时查看上传的裁判文书内容，并且可以点击查看推荐的相似裁判文书内容。

图3-1 系统用例图

相似文书推荐，是该系统的关键。该需求是当用户上传当前裁判文书文档之后，根据系统算法计算结果，系统在语料库中找到和当前裁判文书相似度最高的案件，并推荐给法官供其参考。同时相似文书推荐又可细分为三个潜在需求：语料库建立，主要是利用爬虫尽可能多的爬取裁判文书，并进行预处理，；文书关键信息抽取，利用相关算法，建立向量模型，提取关键词；相似度计算，利用余弦相似度算法，计算两个裁判文书的相似度。

3.1.2 非功能性需求

准确率：中文分词的准确率达到95%以上，中文分词的高准确率是相似度算法的基础。如果无法正确划分中文词语，关键信息将无法提取。同时，相似度计算的准确率应高于单一TFIDF计算结果。

可靠性：系统应保持长期稳定运行，并保证输出结果的稳定及准确。

便捷性：系统应易于部署。

3.2 项目总体设计

图3-2 系统总体架构

如图3-2所示，基于强化学习的案件相似度计算系统包含以下几个模块：裁判文书抓取模块、裁判文书处理模块、模型训练模块、相似度计算模块和Web工程模块。

其中，裁判文书抓取模块主要负责利用Python爬虫在网络上抓取案件裁判文书并将裁判文书保存至文书库。而抓取来的大量文书则进一步交由裁判文书处理模块处理。在这一模块中，首先进行裁判文书切分处理，按照结构分段标记，随后将其核心段落交由JIEBA分词系统进行中文分词处理。模型训练模块负责利用TFIDF算法计算每个分词的TF*IDF值，并利用强化学习算法过滤冗余词语。随后将处理后的数据经序列化处理保存到模型文件。相似度计算模块的作用是利用向量模型中的数据，利用余弦相似度算法，计算案件相似度，并按照相似度降序排列。最后选择相似度最高的案件文书作为返回结果展示给用户。而Web工程模块对外负责UI页面的用户交互，包括裁判文书的上传和相似裁判文书的选择等，对内负责数据存取和控制调用各个模块。

所以，该系统是一个分层架构，数据在各个模块依次流转，各个模块相互独立，又环环相扣。

3.3 项目模块设计

3.3.1 裁判文书抓取模块

裁判文书抓取模块的功能主要是从互联网上各个裁判文书网站爬取尽可能多的裁判文书。经多方查阅，中国裁判文书网上的裁判文书种类较全，数量众多，可以作为裁判文书主要来源。该模块本质上是一个网络爬虫，需要注意以下几点：

1. 爬虫操作频率不可过快，否则会导致目标网站的崩溃。

2. URL去重是爬虫运行中一项关键的步骤，由于运行中的爬虫主要阻塞在网络交互中，因此避免重复的网络交互至关重

要。一般会将待抓取的URL放在一个队列中，从抓取后的网页中提取到新的URL，在他们被放入队列之前，首先要确定这些新的URL没有被抓取过，如果之前已经抓取过了，就不再放入队列了[]。

3. 默认情况下，Request 将无限期地一直等待响应。因此，建议设置超时参数。

该模块包含Download、DownloadManager、ContentSave类。其中Download类负责网页内容的抓取；DownloadManager负责控制Download，传递参数等；ContentSave类负责将抓取的网页内容按照结构存入数据库及txt文件中。

3.3.2 裁判文书处理模块

裁判文书处理模块有两大主要功能:裁判文书的切分处理和裁判文书的中文分词。模块的的输入是文书库中的裁判文书原文件，输出是标记好分段，并分词处理过后的裁判文书文件。

裁判文书看似冗长而繁杂无章，实则却有着固定的格式，且包含着重要信息。因此裁判文书的切分处理十分必要。通过观察分析裁判文书结构可知，裁判文书相似度对比主要涉及的就是案由部分的对比。同时，后续的裁判文书中文分词处理以及模型训练模块，相似度计算模块都是对裁判文书案由等部分的操作。因此准确的裁判文书的切分处理是整个裁判文书相似度计算算法的基础。

本系统按照：法院名称、文书名称、案号、首部、事实、理由、裁判依据、裁判主文、尾部、署名、日期十条规则将裁判文书划分为十个部分并加以标记，以供后期建立模型时给用户可选择的评判标准。

裁判文书的切分处理功能由split_case.py实现，主要包含四个方法：get_text、find_line_no、cut_case、write2file。其中get_text负责将裁判文书按行读取到列表中。find_line_no负责根据参数传入的关键字在lines中寻找对应的行，如果已知关键字在文件尾部，可将flag设置为False以执行倒序寻找，提高程序效率，降低时间复杂度。cut_case负责按规则切分文本。具体规则需要详细了解裁判文书格式后加以确定。最后，write2file负责将切分后的裁判文书存入指定位置。

中文分词的实现有多种方式，本系统采用jieba中文分词组件来实现案件书的分词工作。将需要分词的案件文书放到同一文件夹下，调用jieba接口即可得到分词后的裁判文书文档。

中文分词功能由jieba_seg.py实现，包含两个方法：getFileList和cut。getFileList方法将读取所有切分后的裁判文书，获取文件名列表。cut则遍历列表，依次打开文件，并调用JIEBA分词接口，将文件中的裁判文书作为参数传入。最终获得分词处理后的文件。

3.3.3 模型训练模块

算法模型的建立主要是将大量的计算任务集中在线下完成，当新的案件文书传入，只需调用建立好的模型，经过少量计算即可快速得出结果。基础的TFIDF算法需要计算语料库中全部词语的TF值和IDF值，模型建立的主要工作便是按照案件文书分类，将每个案件文书中的中文分词的TF、IDF值加以保存。

Train_Model.py负责的就是向量模型的初步建立工作，该模块涉及四个方法：get_text、extract、make_documents和tfidf。其中get_text负责读取文书库中文件，将文件中文书读取到列表中。在文书处理模块中，裁判文书已经被按结构节分并标记，而extract负责根据需要，按关键部分提取文书内容。make_documents负责将文件生成所需格式。tfidf则负责具体的TFIDF值计算。

在向量模型初步建立后，随即调用强化学习算法优化模型。由于利用基于词频的TFIDF算法求得的分词权值存在一定的噪音，本文采用Q-Learnig算法尝试修正这些异常数据。

Q-Learning算法的基础公式如(8)

(8)

其中为一组状态与行为序列，为序列的reward之和，是序列出现的概率，即同时拥有多组轨迹，取均值。

Qlearning.py负责利用强化学习优化向量模型。根据以上算法，本系统设计了以下几个方法：wordFilter、build_q_table、choose_action、get_env_feedback、update_env以及RL。RL是强化学习算法的主方法，包含了公式(8)的具体实现。在RL中首先调用wordFilter方法，将文书库中所有分词按照出现频率排序，并筛选出出现频率最高的词。随后调用build_q_table方法建立Q表，将所有阶段动作Q值初始化为0。算法开始执行后，依次调用choose_action、get_env_feedback、update_env，利用Epsilon-greedy算法选取动作，并获取下一状态及其奖励，最后更新Q表及环境。当算法执行结束后，将训练后的Q表作为结果返回，根据Q表来控制向量模型中数据的筛选和优化工作。

最后便是将优化后的向量模型序列化处理。通过将对象序列化可以将其存储在变量或者文件中，可以保存当时对象的状态，实现其生命周期的延长。并且需要时可以再次将这个对象读取出来。save_model方法负责模型序列化的具体实现，引用了Python标准库中的cPickle模块，通过调用cPickle.dump方法，将处理后的向量模型保存。

3.3.4 裁判文书相似度计算模块

裁判文书相似度计算模块的主要工作是利用余弦相似度公式计算案件相似度。该模块的计算主要依赖模型训练模块的向量模型。向量模型中记录着每个案件文书中每个分词的权值。

余弦相似性，是用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小的度量。余弦值越接近1，就表明夹角越接近0度，也就是两个向量越相似[]。

图3-3 三角形余弦公式示意图

由图3-3可推出三角形余弦定理公式：

（13）

在用向量表显示的三角形中，根据公式(13)三角形余弦定理公式可改写成如下形式：

（14）

利用该原理，结合两个案件文书的分词权值，即可得出两个案件文书的相似度。

裁判文书相似度计算模块由simComp.py实现，该模块较为复杂，涉及方法较多。总体来说分为两部分，新上传文书的处理和相似度计算。新上传文书处理功能的实现涉及到的方法主要有：make_documents、test_tfidf。make_documents负责将新上传的裁判文书进行段落切分和中文分词处理；test_tfidf负责计算新裁判文书中分词的TFIDF值。相似度计算功能涉及到的方法有Process、find_max_sim、similar以及cosine。其中Process是相似度计算的主方法，负责参数处理，控制循环和输出结果；find_max_sim负责控制循环遍历文书库中所有裁判文书，并调用余弦相似度算法，将结果排序返回；similar负责处理参数，并将参数传递给cosine；cosine则负责余弦相似度的具体逻辑计算。

3.3.5 Web工程模块

Web工程模块的主要功能是用户上传新的裁判文书，以及相似文书的推荐展示。该模块基于Python的Django框架，采用MTV的设计思想。

如图3-4所示，当用户在前端添加一个新的裁判文书，浏览器将表单发送给View，View调用相似度计算模块，将裁判文书内容作为参数传入，同时将该文书存入文本库。相似度计算模块通过与语料库中数据的计算，返回相似度最高的N个案件名，交由View返回给前端页面展示。

图3-4 添加新文书时序图

当用户看到展示的相似案件名称后，点进即可查看推荐的案件文书内容。如图3-5户从浏览器发送表单到View请求查看一个裁判文书，View根据文件名调用find方法从文本库中查找裁判文书，并将结果返回前端页面展示。

有关页面交互的部分主要由Django框架的View.py负责，包含了add_content、get_content、handle_uploaded_file三个方法。其中add_content负责用户上传的新的裁判文书，将其保存到文书库，并调用handle_uploaded_file方法。顾名思义，handle_uploaded_file负责处理上传的裁判文书，调用相似度计算算法，返回相似度最高的案件列表。get_content负责根据裁判文书名称从文书库中查找需要的裁判文书，并将结果展示在页面上。

图3-5 查看文书时序图

3.4 本章小结

本章节主要介绍基于强化学习的案件相似度计算系统的分析与设计。首先介绍了项目的主要需求并加以分析，介绍了现有系统的不足和本系统的设计目标。同时给出了系统用例图。然后介绍了项目的总体设计，以及系统的模块划分及各个模块的运作流程。具体包括裁判文书抓取模块、裁判文书处理模块、模型训练模块、相似度计算模块和Web工程模块。在分析了系统需求并介绍了系统总体设计后，详细解释了各个模块的作用以及设计思路，实现方案，包括了各个模块的流程图以及关键技术要点。通过本章的的介绍，可以对基于强化学习的案件相似度计算系统有一个初步的了解。

指　标
疑似剽窃文字表述

1. 以下几个模块：裁判文书抓取模块、裁判文书处理模块、模型训练模块、相似度计算模块和Web工程模块。
2. URL去重是爬虫运行中一项关键的步骤，由于运行中的爬虫主要阻塞在网络交互中，因此避免重复的网络交互至关重要
3. 余弦相似性，是用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小的度量

脚注和尾注

1. [] 爬虫技术浅析[EB/OL]. https://blog.csdn.net/junli_chen/article/details/49591543, 2015..
2. [] 邓琨. 基于邻域的协同过滤推荐系统相似度研究[D]. 江西财经大学, 2017..

5. 第4章基于强化学习的案件相似度计算方法实现

总字数：4854

相似文献列表　文字复制比：2.3%(113)　疑似剽窃观点：(0)

1	50582122786987440_成瑞_面向视频网站的有效评论识别方法的研究与实现 成瑞 -《高职高专院校联合比对库》- 2017-06-03	0.8%（38） 是否引证：否
2	面向视频网站的有效评论识别方法的研究与实现 成瑞 -《大学生论文联合比对库》- 2017-05-30	0.8%（38） 是否引证：否
3	成瑞-12124003-计算机科学与技术 成瑞 -《大学生论文联合比对库》- 2017-05-30	0.8%（38） 是否引证：否
4	面向视频网站的有效评论识别方法的研究与实现 赵月 -《大学生论文联合比对库》- 2017-06-03	0.8%（38） 是否引证：否
5	python中jieba分词快速入门 - Together -《网络（http://blog.csdn.net）》- 2017	0.8%（38） 是否引证：否

第4章基于强化学习的案件相似度计算方法实现

4.1 裁判文书抓取模块

裁判文书抓取模块本质上是一个网络爬虫，采用Python语言结合Selenium工具，将尽可能多的案件文书从互联网下载到本地 。经多方查阅，中国裁判文书网上的裁判文书种类较全，数量众多，可以作为裁判文书主要来源。

Selenium是一个Web应用测试工具，能够模拟人在浏览器的各种操作。Selenium支持各种浏览器驱动，包括谷歌Chrome、IE以及火狐Firefox。本文采用的是Firefox。

如表4.1所示，该模块主要包含以下几个类：DownloadManager，Download，ContentSave。DownloadManager负责管理爬虫类Download，控制爬虫起始地址。由于案件文书分为几大类：刑事案件、民事案件、行政案件、赔偿案件、执行案件等，为了方便控制爬虫爬取的案件类型，本模块采用人工写入的方式将多个爬虫的起始地址写入变量中，根据需要将对应地址作为参数传入。同时爬取文件的存储位置也可在DownloadManager类中手动设计定。由于该模块为线下模块，对于执行速度并不是最主要要求，因此并没有引入多线程，后期版本中可以加以优化。

分类类名

控制类 DownloadManager

爬虫类 Download

存储类 ContentSave

表4-1 裁判文书抓取模块类表

如图4-1所示，案件文书的具体爬取工作主要由Download完成。为了防止访问过快，目标网站崩溃，系统设置了访问等待时间6秒。爬虫通过读取起始页信息，获取案件名称列表，并根据列表依次打开每一个案件详情页。经过观察，裁判文书内容在案件详情页的HTML代码中具有鲜明的ID，因此爬虫使用find_element_by_id方法即可抓取全部文书内容。最后调用ContentSave类中的save方法，将文书内容按文件名一一存入文书库。

图4-1 Download类代码

ContentSave类实现的功能较为简单，即案件文书保存。只需调用该类中的save方法，将文书库位置，案件标题，案件内容作为参数传入，即可便捷实现保存功能。

4.2 裁判文书处理模块

裁判文书处理模块主要实现两个功能，一是裁判文书的切分，二是案件文书的中文分词处理。由于中文分词处理需要频繁网络交互，较为耗时，且分词处理前需提取裁判文书的关键段落，因此本文将这两个功能合并为同一模块，作为建立模型前对裁判文书文件的预处理。

4.2.1 裁判文书的切分处理

通过观察分析裁判文书结构可知，裁判文书相似度对比最重要的就是案由等部分的对比。对裁判文书案由等部分的操作贯穿整个相似度计算算法。因此如何在复杂的裁判文书中将案由等部分标记切分出来，是相似度计算算法实现的重要基础工作。

该功能实现所依赖的一个重要的方法find_line_no，该方法有三个参数lines, s以及flag。该方法根据关键字s在lines中寻找对应的行，如果明确知晓关键字位于案件文书后方位置，可将flag设置为False以实现倒序寻找，减小时间复杂度。

图4-2 原文件内容

借助find_line_no方法，通过关键词查找，该算法根据法院名称、文书名称、案号、首部、事实、理由、裁判依据、裁判主文、尾部、署名、日期，将案件文书分为十部分[]，并加以标记。如图4-2所示为原文件的格式。图4-3所示则为切分处理后的裁判文书格式。

图4-3 切分处理后的文件内容

4.2.2 案件文书的中文分词处理

为了减少建立语料库是的人工工作量，本文决定采用现有的成熟中文分词工具。通过比较哈工大的LTP，NLPIR以及JIEBA分词工具，三者各有优缺点，本文本着简便易用的原则，决定采用JIEBA分词来完成案件文书中文分词处理。

如图4-4所示为中文分词的代码实现，jieba.cut方法接受三个输入参数:一是需要分词的字符串，本系统将处理后的裁判文书文件按行读取，并作为该参数传入；二是cut_all 参数用来控制是否采用全模式，本文采用默认的精确模式，cut_all参数缺省；三是HMM参数用来控制是否使用HMM 模型。同时由于JIEBA分词是面向各个领域具有普适性的分词系统，因此对于特定领域难免有无法识别的分词，这时可以调用jieba.load_userdict方法，添加自定义词典，将词典文件作为参数传入。此时JIEBA分

词系统将更加准确。

通过调用getFileList方法，系统获取到需要进行批量分词处理的文件名列表，并通过循环遍历列表，调用cut方法。cut方法会将裁判文书文件中的内容作为参数，调用JIEBA的API接口，并返回分词后的结果，存入目标文件中。JIEBA分词使用简单，只需通过import jieba来引用即可。

解析到的结果如图4-5所示：

图4-4 中文分词代码

图4-5 中文分词结果

4.3 模型训练模块

模型训练模块的主要任务是裁判文书向量模型的建立。裁判文书在完成切分处理和中文分词操作后，交由该模块进行进行处理。模型建立的过程本质上是Key-Value对的建立，将分词作为key值，将计算后的特征值作为value值。随后利用强化学习算法进行模型优化，提高准确性。

4.3.1 模型初步建立

图4-6 Document类代码

如图4-6所示，Document类作为案件文书的实体类，包含两个成员变量：fillename和words，分别存储文件名和提取后的文件内容。

模型训练模块在train_model.py中得以实现。该模块包含make_documents、tfidf和extract三个主要方法。make_documents方法负责裁判文书的预处理，将裁判文书文件生成所需的格式。首先扫描文本库中所有案件文书，将裁判文书中内容读取到列表中，然后调用extract方法。extract方法负责按关键部分提取文书内容，并统计该部分的词频，计算每个文件中的每个分词出现的次数，并以Key-Value对的形式保存到Document的words变量中。最终将计算后所得的Document数组作为参数，调用tfidf方法。tf值是指一个词语在本篇裁判文书中出现的频率，idf值代表着该词语在语料库其他文章中出现的频率，从统计学的角度上讲，td*idf的值越大，这个词语对该篇裁判文书重要性越高。如图4-7所示tfidf方法在make_documents方法结果的基础上，分别计算每个案件文书中的每个分词的tf值和idf值，并将td*idf的最终结果替换掉对应Key-Value对中的Value值。

图4-7 tfidf方法代码

4.3.2 强化学习算法优化

该模块将文书库中所有分词按出现频率排序，然后将出现频率最高的N个词交由强化学习算法进行筛选。最后根据筛选结果，将建立好的向量模型加以优化。

首先要进行的Q表的建立。如图4-8所示，Q表实质上是一个二维表格，记录着分词和动作选项。YES代表该分词得以保留，NO代表删除该分词。

图4-8 Q表建立代码

本系统根据公式(8)实现了Q-Learning算法。如图4-9所示，每次更新Q表时，我们都用到了Q_target值和Q_predict值。将当前状态的估计值乘上衰减率，加上当前得到的奖励，就是当前状态的现实值。

如图4-10所示为动作选择算法。在该算法中，运用了Epsilon greedy决策方法。Epsilon是一个概率，在本系统中，将Epsilon设置为0.9。此时有九成的可能选择Q表中Q值较大的动作，而只有一成的可能会随机选择一个动作。Q_target值与Q_predict值的差是误差，Alpha是一个小于1的效率值，用误差乘以Alpha则得到本次学习到的Q值。gamma是对未来reward的衰减值。

经过训练，我们得到了最终的Q表，根据该Q表，我们可以确定向量模型中哪些分词可以被保留，哪些分词需要删除。最终得到我们所需要的优化后的向量模型。

图4-9 Q-Learning算法代码

图4-10 动作选择算法代码

4.3.3 向量模型序列化

图4-11 向量模型序列化代码

最后便是建立完毕后模型的序列化。Python序列化的概念很简单：通过将对象序列化可以将其存储在变量或者文件中，可以保存当时对象的状态，实现其生命周期的延长。并且需要时可以再次将这个对象读取出来。简单地说就是一个捕获了当前进度的数据结构预先保存到硬盘上，接着在你重新需要使用的时候从硬盘上加载进来[]。

如图4-11所示，本文引用的是Python标准库中的cPickle模块，通过调用调用cPickle.dump，将处理后的向量模型保存到名为model.pkl的模型文件中，供相似度计算模块使用。

4.4 相似度计算模块

在向量模型建立后，根据模型中的数据，每篇裁判文书信息都被抽象成了向量。根据公式(11)所示的余弦相似度计算公式，即可开始两篇裁判文书的相似度计算。

（11）

如图4-12所示，find_max_sim负责控制最外层循环遍历文书库中所有文书，将其与法官新上传的文书对比。similar负责参数的处理，并将参数传递给cosine开始进行余弦相似度计算。

图4-12 相似度计算代码

cosine方法中，numerator即余弦相似度计算公式中的，两篇文章中同时出现的关键词权值相乘。sourceLen即，计算第一篇文章的向量长度。同理targetLen即，计算第二篇文章的向量长度。denominator即sourceLen和targetLen的乘积，通过求numerator/denominator的值，即可得出余弦相似度的最终结果。

4.5 Web工程模块

本系统的Web服务框架基于Python的Django框架搭建。Django框架采用了MTV的框架模式，即模型M，模板T和视图V。

该系统在Django框架下只有一个模型，即Case。该模型用以保存裁判文书内容和标题等信息。视图有着承上启下的作用。在view.py中，有三个主要方法：add_content、find_sim和show。这三个方法分别用以上传文书文件，查找相似度最高的案件和查看案件。模板，主要用于展示业务逻辑，通常是HTML网页的形式。该系统主要有3个模板，分别用以展示起始页，相似文书推荐页和案件详情页。

Django框架使用url.py配置请求的处理函数。当用户选择了文书文件，点击上传按钮时，调用add_content方法，将案件文书存入文本库，随后系统调用find_sim方法，进行裁判文书的数据处理和相似度对比工作，并将计算结果返回到前端页面供用户选择。当用户点击文件名，则调用show方法，在文本库中查找该文件，并展示最终结果。

4.6 本章小结

本章重点介绍了基于强化学习的案件相似度计算方法的具体实现。首先介绍了裁判文书的爬取模块，解释了如何利用Python爬虫，将中国裁判文书网的裁判文书按文书结构和类型保存至文书库。随后介绍了裁判文书的初步处理，解释了如何将裁判文书按照段落内容特征进行切分处理，并介绍了如何利用JIEBA分词系统进行裁判文书的中文分词处理。接下来，介绍了相似度算法向量模型的建立。首先利用TFIDF算法计算裁判文书中所有分词的TFIDF值，随后利用强化学习算法，将分词进行过滤和优化。最后将处理后的数据将计算后的案件信息经序列化保存至Model.pkl文件。随后介绍了最重要的相似度计算模块。解释了在该模块中如何处理新上传文书，并读取向量模型文件。介绍了余弦相似度算法的实现，以及最终相似案件推荐结果的取得。最后概述了该相似度计算方法在Django框架下的实际部署。

脚注和尾注

1. [] 杨翔, 奉鑫庭. 中国民事裁判说理:路径选择与实现方式[J]. 湘潭大学学报(哲学社会科学版), 2017, 41(5):83-88..
2. [] Sun Haiyu. Python序列化和反序列化[EB/OL]．https://www.cnblogs.com/sun-haiyu/p/7087088.html..

6. 第5章系统测试

总字数：1683

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第5章系统测试

本文使用的开发平台为Intel(R) Core(TM) i7-6700HK CPU @ 2.60GHz，系统内存(RAM)为16.00GB(15.9GB可用)，Windows 10系统。使用Python2.7作为程序实现语言，开发工具为Pycharm。

裁判文书种类繁多，本文选取婚姻裁判文书作为测试用例。通过通过裁判文书爬取模块，已预先在中国裁判文书网上爬取婚姻裁判文书2824篇。本章将利用文书库中婚姻裁判文书进行基于强化学习的案件相似度计算方法的系统测试。

5.1 案件相似度计算系统演示

图5-1 案件相似度计算系统首页

如图5-1所示为案件相似度计算系统的操作页面，点击选择文件，将裁判文书上传至系统，点击提交，自动调用相似度计算算法，并跳转到如图5-2所示页面。该页面展示了经案件相似度计算后的相似裁判文书推荐结果。

图5-2 案件相似度计算系统结果页

通过点击对应的裁判文书名，跳转到如图5-3所示页面。该页面展示了裁判文书的具体内容。并可以以返回上层或首页继续执行操作。

图5-3 案件相似度计算系统文书详情页

5.2 裁判文书处理模块测试

图5-4 裁判文书切分处理

首先将需要切分处理的2824篇婚姻裁判文书放入hunyin_case文件夹中。启动cmd命令行，输入命令：

python split_case.py hunyin_case split_case

运行结果如图5-4所示。运行时间9.004秒，切分案件文书2824篇。处理后的裁判文书文件存入了split_case文件夹，经手工检查，段落切分结果正确。

随后进行中文分词处理，输入端是切分处理后的split_case文件夹，输出端是split_seg_case文件夹。启动cmd命令行，输入命令：

python jieba_seg.py split_case split_seg_case

运行结果如图5-5所示，运行时间107.44秒。

图5-5 裁判文书中文分词处理

5.3 案件相似度计算模块测试

本小节随机选取了100篇案件文书分十次进行案件相似度计算。将目标裁判文书放入test_path文件夹中，调用命令：python cal_similarity.py test_path split_seg_case 3。系统运行结果如图5-6所示。经10次运行，系统运算时间记录如表5-1所示。

实验编号 1 2 3 4 5 6 7 8 9 10

运行时间 1.14s 1.18s 1.19s 1.08s 1.17s 1.20s 1.11s 1.18s 1.14s 1.21s

表5-1 裁判文书相似度计算结果

经计算，系统十次运算，每次处理10个案件，平均运行时间1.16秒。响应速度达到了设计要求。经人工检查，推荐的相似案件准确率较高，符合预期。

图5-6 案件相似度计算模块测试

5.4 性能对比

图5-7 案件相似度计算模块测试

如图5-7所示，5次实验中，本系统对于裁判文书相似度的计算准确性还是比较理想的。相较于传统基于词频的TFIDF,基于强化学习的案件相似度计算系统虽然在线下建立向量模型的速度上有一定牺牲，但在裁判文书相似度判断准确性有了一定提高，达到了系统设计目的。

5.5 本章小结

本章首先进行了系统演示，在上传裁判文书后，系统成功展示了相似裁判文书推荐列表，并根据相似度进行了排序，随后点击裁判文书名，可以查看裁判文书的具体内容。系统响应较为迅速，结果准确。

随后进行了系统性能的测试。分别测试了裁判文书处理模块和案件相似度计算模块。两个模块运行顺利，结果准确。时间效率上裁判文书处理模块由于涉及网络传输，虽然运行时间较长，但并未超出预期；案件相似度计算模块响应迅速，运行稳定。

最后对比了该算法和传统基于词频的相似度计算方法的效率，可以看出经过强化学习算法优化后的相似度计算结果具有更高的准确率。同时可以看出有更具专业性的案件文本库对于提高相似度计算准确性有着重要作用。

7. 第6章总结与展望

总字数：1526

相似文献列表　文字复制比：1.9%(29)　疑似剽窃观点：(0)

1	农村电商运营系统项目风险管理 张丹(导师：卢子芳) -《南京邮电大学博士论文》- 2016-09-12	1.9%（29） 是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第6章总结与展望

6.1 总结

本文提出了基于强化学习的案件相似度计算方法。首先介绍了项目的背景和已有的相关技术与研究。随后介绍了本系统涉及到的相关技术及算法，如结巴分词系统，Q-Learning、TFIFD算法等。随后提出了基于强化学习的案件相似度计算方法的设计，分析了系统需求，并介绍了项目总体规划以及各个模块的设计思路。然后介绍了基于强化学习的案件相似度计算方法的实现，详细解释了各个模块的实现方法和技术要点。最后系统测试，先演示了案件相似度计算系统，随后测试了裁判文书处理模块和案件相似度计算模块的性能并对比了该系统和传统基于词频的相似度计算方法的效率。

本文使用了自然语言处理技术来进行案件裁判文书的分词处理，并利用TFIDF算法初步建立了向量模型，随后使用强化学习算法优化选择关键词，提高关键词准确性。最后利用余弦相似度算法，计算出相似度最高的裁判文书，并推荐给法官，供其在审判案件时做出参考。在本系统中，主要工作分为四大部分：裁判文书的抓取，裁判文书的预处理，向量模型的建立以及裁判文书的相似度计算。裁判文书抓取模块通过Python爬虫从中国裁判文书网抓取大量裁判文书，并根据裁判文书结构和类型存入文书库。裁判文书处理模块加载文书库中的文件，进行段落切分处理以及中文分词处理。随后利用TFIDF算法结合强化学习算法，优化筛选关键词，并利用处理后的数据建立向量模型，将向量模型序列化并保存。最后，根据余弦相似度算法计算出两篇裁判文书的相似度并按降序排列。最终根据需要，将相似度最高的N篇裁判文书推荐并展示。

相比于传统的全文检索和基于词频的关键词提取算法，本系统的裁判文书相似度计算结果具有较高的准确率，系统响应速度较快且运行稳定。该系统可以更准确地根据用户需求，乃至其潜在需求，找出相似的裁判文书，并加以展示。对于法官办公效率的提高和审判质量的提升有着更好的辅助效果。

6.2 工作展望

本文实现的基于强化学习的基于强化学习的案件相似度计算方法，是强化学习算法和自然语言处理系统在法院裁判文书领域的一次简单应用。该系统综合运用了各方面的知识，具有一定的示范作用，并且对于其他领域，如新闻报道搜索，图书管理，博客、论坛等文章检索等有着很大的扩展性。只需找到特定领域的语料库，并使用恰当的机器学习算法，相信同样可以取

得准确高效的结果。

本文实现的系统，客观地看还存在一定的不足和可以进一步优化的地方。首先，Python爬虫算法的设计还处于最简易的版本，采用单线程，效率不高。由于爬虫的效率很大程度上取决于网络传输，因此多线程是一个很好的优化方向。其次，由于裁判文书有多种类型和不同格式，本文只选择了婚姻案件作为最初版本的实验样本。在后续版本中，可以加入更多文书类型，扩大文书库，以提供更具普适性的相似案件文书推荐系统。最后在相似度计算问题上，本文采用关键词余弦相似度作为计算结果，可以引入多维度多角度的判定标准，如参考的法律条文，案由等是否一致。这就需要对案件文书和相关法律知识有着更深的了解，才能取得更好的效果。

参考文献

致谢

我首先要感谢我的论文指导老师，吉林大学软件学院的张睿老师。张睿老师在毕业研究方向的确定上给了我指导性推荐。在论文撰写过程中，也耐心的为我答疑结果，在遇到技术瓶颈时给出建设性的意见和帮助。同时，还要感谢吉林大学计算机科学与技术学院的授课老师们和所有同学们，在四年的大学时光里相互扶持度过了难忘的美好时光。

此外，还要感谢我实习公司的同事们，在论文撰写过程中给我技术和业务上的帮助和支持，使我的研究课题有了切实的理论支持，并帮助我对课题有了更深的了解。

最后，感谢论文评阅老师们的辛苦工作。