南京大學
NANJING UNIVERSITY

研 究 生 毕 业 论 文

（申请硕士专业学位）

论 文 题 目 司法文本数据自动化生成系统的设计与实现

作 者 姓 名 张舒

专 业 名 称 工程硕士（软件工程领域）

研 究 方 向 软件工程

指 导 教 师 陈振宇 教授

2020 年 5 月 19 日

学　　号 ：MF1832243

论文答辩日期 ： 2020 年 5 月 21 日

指 导 教 师 ：（签字）

The Design and Implementation of Judicial Text Data Automatic Generation System

By

Shu Zhang

Supervised by

Professor **Zhenyu Chen**

A Thesis
Submitted to the Software Institute
and the Graduate School
of Nanjing University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Engineering

Software Institute

May 2020

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：司法文本数据自动化生成系统的设计与实现

工程硕士（软件工程领域）专业 2018 级硕士生姓名：张舒

指导教师（姓名、职称）：陈振宇 教授

摘 要

在我国智慧法院的建设背景下，司法机关纷纷将深度学习引入司法领域及法律服务领域。司法深度学习模型大多数以裁判文书或案情事实等司法文本作为输入，研究比较广泛的类型有自动量刑预测系统、法条及罪名预测系统、类案（相似案件）推荐系统等。文本数据的缺乏会对司法深度模型表现产生负面影响，在模型训练阶段，存在训练数据缺乏导致模型泛化能力差的现象；在测试阶段，存在测试指标单一化现象，缺乏结合司法行业特性设计的具有多维度测试功能的测试数据集。

本文设计并实现了司法文本数据自动化生成系统，分为训练数据生成模块与测试数据生成模块。训练数据生成模块用于为司法深度学习模型提供数据扩增服务，增加高质量的司法文本训练数据，提高模型预测准确率。模块设计了基于规则与基于变分编码器两种生成方式。其中，基于规则的生成方式结合司法文本特性提出了针对于司法文本的扩增方式；基于变分编码器的生成方法将变分编码器应用在文本生成领域，学习高斯分布到数据分布的映射关系，重建具有相似分布的新文本。

测试数据生成模块用于为司法深度学习模型的多维度评价提供测试数据，使模型的测试指标多元化。模块设计了含噪声项测试数据生成方法，用于评价深度学习模型的抗噪能力；设计了对抗攻击测试数据生成方法，通过对测试数据尽可能小的文本改动，对深度学习模型进行基于遗传算法的对抗攻击，经过对抗攻击的测试数据用于评价深度学习模型的抗对抗攻击能力。

系统以基于 Django 框架的 web 应用为载体，支持用户自定义生成参数，生成的文本以文件形式返回给用户。系统使用 HDFS 作为文件管理系统，使文件存储具有可拓展性。实验证明，系统提供的训练数据生成方式可以使以 FastText、TextCNN、LSTM 为结构的罪名预测系统的准确率得到提升。系统提供的两种测试数据生成方式可以支持司法深度学习模型的多维度评估。

关键词：数据生成，自然语言处理，变分编码器，对抗攻击，Hadoop

南京大学研究生毕业论文英文摘要首页用纸

THESIS: The Design and Implementation of Judicial Text Data Automatic Generation System

SPECIALIZATION: Software Engineering

POSTGRADUATE: Shu Zhang

MENTOR: Professor Zhenyu Chen

Abstract

In the context of the construction of smart courts in China, the judiciary has introduced deep learning into the field of justice and legal services, continuously making new achievements in the field of justice and legal services. In the field of natural language processing, judicial deep learning systems use judicial texts such as judgment documents or facts of the case as input. The types of systems that are widely used include automatic sentencing prediction systems, laws and crimes prediction systems, recommendation systems for similar cases and so on. The lack of text data will negatively affect the performance of the in-depth judicial model. In training phase, there is a phenomenon that the lack of training data leads to poor model generalization ability. In testing, the test indicators are singularity, lacking of multi-dimensional test data sets designed with judicial industry characteristics.

This thesis designs an automated system for generating judicial text data and implements it, including training and testing data generation module. The training data generation module is used to provide data augmentation services for the judicial deep learning model, increasing the high-quality judicial text training data and improving the prediction accuracy of the model. We design two generation methods based on rule and Variational Auto-encoder. Combining with the characteristics of judicial text, we propose an augmentation method for judicial text in rule-based generation. The generation method based on Variational Auto-encoder applies the Variational Auto-encoder in the field of text generation, learning the mapping relationship between Gaussian distribution and data distribution and reconstructing new text with similar distribution.

The testing data generation module is used to provide testing data for the multi-dimensional evaluation and diversify the test indicators of the judicial deep learning

model. The module also designs a method of generating testing data with noise items, which is used to evaluate the anti-noise ability of the deep learning model. The module also designs a method of generating testing data for adversarial-attack. The deep learning model suffers fro adversarial-attack based on genetic algorithm by changing the text as little as possible. Then the testing data after the adversarial-attack is used to evaluate the anti-attack ability of the deep learning model.

The system uses a web application based on the Django framework as a carrier. The system support user-defined generation parameters, and the generated text is returned to the user with a file. The system uses HDFS as a file management system to enhance the scalability of file storage. This thesis conducted extensive experiments to prove that the two training data generation methods provided by the system are effective, and can increase the accuracy of the crime prediction model based on FastText、TextCNN、LSTM. The two testing data generation methods provided by the system can support multi-dimensional evaluation of judicial deep learning models.

Keywords: Data Generation, Natural Language Processing, Variational Auto-encoder, Adversarial Attacks, Hadoop

目录

表 目 录 ····· **viii**

图 目 录 ····· **x**

第一章 绪论 ····· **1**

1.1 选题背景与意义 ····· 1

1.2 国内外研究现状及分析 ····· 2

1.2.1 文本数据扩增研究现状 ····· 2

1.2.2 深度学习模型多维度测试研究现状 ····· 3

1.3 本文主要研究工作 ····· 4

1.4 本文的组织结构 ····· 5

第二章 技术综述 ····· **6**

2.1 自然语言处理技术 ····· 6

2.1.1 中文分词技术 ····· 6

2.1.2 词向量技术 ····· 7

2.1.3 语言模型 ····· 8

2.2 变分编码器 ····· 9

2.3 Django 框架 ····· 11

2.4 遗传算法 ····· 12

2.5 分布式存储技术 ····· 14

2.6 本章小结 ····· 15

第三章 系统需求分析与架构设计 ····· **16**

3.1 系统需求分析 ····· 16

3.1.1 数据交互需求分析 ····· 17

3.1.2 文本预处理需求分析 ····· 18

3.1.3 数据生成需求分析 ····· 18

3.1.4 用户文件管理需求分析 ····· 21
3.1.5 非功能性需求分析 ····· 21
3.2 系统架构设计 ····· 22
3.3 数据交互模块设计 ····· 27
3.4 文本预处理模块设计 ····· 29
3.5 数据生成模块设计 ····· 31
3.6 用户文件管理模块设计 ····· 34
3.7 数据库设计 ····· 35
3.8 本章小结 ····· 40

第四章 系统实现 ····· 42
4.1 文本预处理 ····· 42
4.2 数据生成 ····· 43
4.2.1 基于规则的训练数据生成 ····· 43
4.2.2 基于 VAE 的训练数据生成 ····· 45
4.2.3 含噪声项测试数据生成 ····· 49
4.2.4 对抗攻击测试数据生成 ····· 50
4.3 数据交互 ····· 53
4.4 用户文件管理 ····· 55
4.5 案例及实验分析 ····· 57
4.5.1 训练数据生成案例及实验分析 ····· 57
4.5.2 测试数据生成案例及实验分析 ····· 61
4.6 系统测试 ····· 63
4.6.1 测试环境 ····· 63
4.6.2 功能测试 ····· 63
4.6.3 性能测试 ····· 68
4.7 本章小结 ····· 68

第五章 总结与展望 ····· 70
5.1 总结 ····· 70
5.2 进一步工作 ····· 70

参考文献 ····· **72**

简历与科研成果 ····· **77**

致谢 ····· **78**

版权与原创性说明 ····· **79**

表 目 录

2.1 Django 的 MTV 模式功能 ····· 11

3.1 数据交互用例状态描述表 ····· 18
3.2 文本预处理用例状态描述表 ····· 19
3.3 基于规则的训练数据生成用例状态描述表 ····· 19
3.4 基于 VAE 的训练数据生成用例状态描述表 ····· 20
3.5 含噪声项测试数据生成用例状态描述表 ····· 20
3.6 对抗攻击测试数据生成用例状态描述表 ····· 21
3.7 用户文件管理用例状态描述表 ····· 22
3.8 文件信息表 File 表结构 ····· 38
3.9 用户信息表 User 表结构 ····· 38
3.10 文件链接表 Link 表结构 ····· 38
3.11 文件集合表 Models 表结构 ····· 39
3.12 具体模型信息表 Case 表结构 ····· 39
3.13 图片信息表 Figure 表结构 ····· 40
3.14 省份信息表 Province 表结构 ····· 40
3.15 城市信息表 City 表结构 ····· 40
3.16 区县信息表 District 表结构 ····· 40

4.1 文本预处理结构化文本格式 ····· 42
4.2 Blind 扩增方法标签说明 ····· 44
4.3 VAE 模型可调整超参数 ····· 47
4.4 对抗攻击测试数据生成中可调整参数 ····· 53
4.5 基于规则的训练数据生成文本示例 ····· 58
4.6 VAE 生成模型的输入数据详情表 ····· 58
4.7 VAE 生成文本范例 ····· 59
4.8 加入扩增数据前后模型预测准确率对比表 ····· 60
4.9 含噪声项测试数据生成文本示例 ····· 61

4.10 原始测试集与含噪声项测试集实验对比表 ····· 62
4.11 对抗攻击模型信息及对抗攻击情况 ····· 62
4.12 对抗攻击测试数据生成文本示例 ····· 63
4.13 系统测试环境 ····· 64
4.14 数据上传测试用例 ····· 64
4.15 文件下载测试用例 ····· 65
4.16 文本预处理测试用例 ····· 65
4.17 基于规则的训练数据生成测试用例 ····· 66
4.18 基于 VAE 的训练数据生成测试用例 ····· 66
4.19 含噪声项测试数据生成测试用例 ····· 67
4.20 对抗攻击测试数据生成测试用例 ····· 67
4.21 用户文件管理测试用例 ····· 68
4.22 系统各生成方式性能测试结果 ····· 68

图 目 录

2.1 FastText 词向量模型结构图 ····· 8
2.2 VAE 训练过程示意图 ····· 11
2.3 遗传算法流程图 ····· 13
2.4 HDFS 模块关系图 ····· 14

3.1 司法文本数据自动化生成系统用例图 ····· 17
3.2 司法文本数据自动化生成系统整体框架图 ····· 23
3.3 司法文本数据自动化生成系统逻辑视图 ····· 24
3.4 司法文本数据自动化生成系统开发视图 ····· 25
3.5 司法文本数据自动化生成系统进程视图 ····· 26
3.6 司法文本数据自动化生成系统物理视图 ····· 26
3.7 司法文本数据自动化生成系统功能模块图 ····· 27
3.8 数据交互模块类图 ····· 28
3.9 系统存储文件流程图 ····· 29
3.10 系统删除文件流程图 ····· 29
3.11 文本预处理模块类图 ····· 30
3.12 文本预处理模块流程图 ····· 31
3.13 数据生成模块功能图 ····· 31
3.14 数据生成模块类图 ····· 32
3.15 基于规则的训练数据生成活动图 ····· 33
3.16 基于 VAE 的训练数据生成活动图 ····· 34
3.17 对抗攻击测试数据生成活动图 ····· 35
3.18 用户文件管理活动图 ····· 36
3.19 系统文件管理数据库实体关系图 ····· 36
3.20 系统 VAE 模型管理数据库实体关系图 ····· 37
3.21 系统地理位置数据库实体关系图 ····· 38

4.1 文本结构化处理核心代码 ····· 43

4.2 语料库词典映射核心代码 ····· 43
4.3 地名替换核心代码 ····· 45
4.4 VAE 文本生成模型训练结构图 ····· 46
4.5 VAE 文本生成模型生成过程结构图 ····· 47
4.6 VAE 结构核心代码 ····· 48
4.7 VAE 模型管理界面 ····· 48
4.8 含冗余项测试数据生成流程图 ····· 49
4.9 含冗余项测试数据生成核心代码 ····· 50
4.10 数据交互实现逻辑 ····· 54
4.11 系统生成结果展示界面示例 ····· 55
4.12 HDFS 设计架构图 ····· 55
4.13 hdfs.xml ····· 56
4.14 core.xml ····· 56
4.15 用户文件管理系统界面 ····· 57
4.16 VAE 生成模型训练过程中 Loss 值与 PPL 值变化图 ····· 59
4.17 TextCNN 预测模型训练过程中准确率值变化图 ····· 61

第一章　绪论

1.1　选题背景与意义

随着近年来大数据与人工智能等技术的高速发展，国内外司法机关纷纷将人工智能引入司法领域及法律服务领域 [1] [2]，司法智能应用系统从基于规则的司法专家系统过渡到基于大数据的司法人工智能系统，在辅助法官办案、服务律师、社会公众方面正在不断取得新的成就 [3]。在自然语言处理领域，司法人工智能系统以裁判文书或案情事实等司法文本作为输入，应用比较广泛的系统类型有自动量刑预测系统 [4]、法条及罪名预测系统 [5]、类案（相似案件）推荐系统 [6] 等。

在司法领域深度学习模型的开发过程中，为了提高模型的表现，一方面要提高模型的识别准确率，另一方面要结合行业特性，提高模型在其他维度上的指标。文本数据的缺乏会对司法深度模型表现产生负面影响。深度学习系统的开发过程包括模型训练与模型测试两个阶段。在司法深度学习模型训练阶段，存在训练数据缺乏导致模型泛化能力差的现象 [7]，模型识别率有待提高；在测试阶段，存在测试指标单一化现象，缺乏结合司法行业特性设计的具有多维度测试功能的测试数据集。

对于司法深度学习系统在训练过程中存在的数据缺乏问题，也普遍存在于其他类型的深度学习系统中。深度神经网络的训练过程需要大量数据，在低数据状态下，参数不确定，学习网络的泛化能力很差。利用数据扩增的手段可以生成一批与原数据具有相同分布的训练数据 [8]，可以更有效地使用现有数据，缓解神经网络泛化能力差的情况，进而提高系统预测准确率。

衡量深度学习系统质量的测试指标除预测准确率外，还应与实际业务相结合，根据实际业务定义测试指标。在司法深度学习领域中，由于模型的输入对象可能是非法律从业者给出的不标准文字表述，因而我们应关注模型的抗噪声能力。由于司法领域深度学习模型的识别结果具有较大的社会价值，可能会对司法领域业务的判定结果产生较大的影响，因此需要关注模型受到黑盒对抗攻击时的表现，即模型的抗对抗攻击能力。对于以上测试指标的需求，可以分别构建含噪声项的、经过对抗攻击的测试数据，根据模型在测试数据上的表现得到对应的测试指标。

综上所述，以文本为输入的司法领域深度学习系统，对训练数据和多维度的测试数据都有较大的需求。对于训练数据的需求，可以通过对已有的训练数

据进行数据扩增，以生成大量具有相同分布的新训练数据；对于多维度测试数据的需求，可以通过对已有的测试数据进行变更与修改，以生成具有定义测试指标能力的新测试数据。本文将训练数据生成与测试数据生成方法集成在一个统一的数据生成系统中，为司法深度学习模型开发者提供更广泛的训练数据与测试数据，辅助模型开发者提高模型表现，具有较强的实用意义。

1.2　国内外研究现状及分析

司法领域文本数据的生成主要分为两个部分，分别为对训练数据的数据扩增操作、与多维度测试数据的生成过程。因此分别调研了目前已有的对于文本的数据扩增的研究进展，与多维度测试数据生成的研究进展。

1.2.1　文本数据扩增研究现状

深度神经网络的训练过程需要大量数据，其训练数据越多，泛化性就越高。数据量少会导致训练的模型难以对测试集中的数据进行泛化，因此模型将存在过拟合的问题。减少过拟合的方法包括在权重范数上添加正则化项、dropout [9]、批标准化（Batch Normalization） [10] 等。而增加训练数据是减轻模型过拟合问题的最根本、最直接的方法。

在可获取的数据较少的情况下，我们可以利用数据扩增的方法，对数据进行扩充，以提高模型的性能 [8]。数据扩增技术指利用可获取的训练数据，使用特定的方法与技巧，增加训练样本的数量以及多样性，数据扩增仅使用训练数据中的信息来增加训练数据的数量。数据扩增技术在图像领域应用较为广泛，在图像领域的传统扩增技术是对原始图像的放射变换，包括图像的旋转、裁剪、调色、以及添加高斯噪声数据等操作 [11]。在近年来，随着生成式对抗网络技术的逐步成熟，H. Lee 等人提出了利用生成式对抗网络技术对图像进行生成来达到数据扩增目的的方法 [12]，A. H. Jha 等人利用变分编码器生成图像，达到数据扩增的目的 [13]。

图像像素点是一种连续的信号，因此图像领域的数据扩增方法通过细微改变像素点的值就可以达到数据扩增的目的。相比较而言，自然语言由词组成，而词语是一种离散的特征，不可能通过细微改变达到不易察觉的变化，因此用于自然语言处理领域的数据扩增较于图像领域受到了较大的限制 [14]。在自然语言中很难获得通用的转换规则，以确保所生成数据的质量，并自动应用于各种行业领域。已有的文本数据扩增方式包括回译法、同义词替换、去信息、改变句法等。回译法指利用翻译工具将句子翻译为其他语言，再翻译回原始语言而生成新数据 [15]。同义词替换指随机选择文本中的单词，将其替换为在词向量中与

其向量距离相近、相似性较高的单词 [16]。去信息指对输入文本随机挑选一部分词做删除操作，每个句子可生成多个具有缺失词语的新句。改变句法方式包括扩句、缩句。缩句指通过句法分析，提取句子的主要结构（例如主语、谓语、宾语部分）形成新句；扩句指通过添加状语、定语等修饰性词语对句子进行扩写。去信息、改变句法等方法在应用时会改变语句的语法结构，很难确保所生成数据的质量，不适合应用于法律行业等语法结构固定的领域。

经过调研发现，目前缺乏为司法领域文本数据设计的数据扩增规则。本文将现有的数据扩增方式结合司法文本的语法特性，通过改变司法文本中某些特征性词语，达到数据扩增的目的。变分编码器技术已经广泛应用于文本数据的生成中，例如自动问答等领域。其生成特点符合数据扩增的要求，因此我们将变分编码器应用于文本数据扩增领域中，在无监督方式下学习数据分布，生成较于基于规则方法更具变化性的数据。

1.2.2　深度学习模型多维度测试研究现状

在对深度学习预测系统进行测试时，系统预测准确率为最基础的测试指标。由于深度学习模型属于黑盒模型，其预测过程是根据模型在大量数据中学习到的统计规律与决策逻辑所做的大规模运算，对于结果的可解释性较差，其行为特性的可理解性较差。结合行业特性设计的测试指标结合深度学习模型的使用场景，根据使用场景提出检测模型鲁棒性的指标，在模型测试过程中增强对模型的解释性与可理解性。

L. Ma 等人提出了针对深度学习模型的 DL 突变测试方法 [17]，其普适于各深度学习系统的测试。除了普适应的模型评价指标，模型的多维度测试应结合所应用行业的行业知识与需求，提供具有行业特性的测试指标。因此，本文结合司法行业特点，设计了多维度测试指标。

W. Hsu 等人指出在许多机器学习场景中，训练和测试之间的分布不匹配会导致性能显著下降 [18]，应用场景中的数据含噪声会使鲁棒性较弱的模型表现下降。因此本文设计了含噪声测试数据的构造方法，测试模型在含噪声数据集上的准确率，用于定义模型的抗噪声能力。

B. Liang 等人指出深度学习文本分类器也容易遭受对抗样本攻击 [19]。当模型受到对抗攻击时，在文本语义改变微小的情况下，模型识别结果会表现出与事实相反的状态。由于司法领域判定结果的社会影响性较大, 因此本文设计了对抗攻击测试数据生成方法，利用生成的对抗攻击数据度量文本分类器被对抗攻击成功的难易程度，即抗对抗攻击能力。

1.3　本文主要研究工作

随着大数据与人工智能技术的飞速发展，各行业领域将大数据与人工智能技术融入行业服务中，以提升行业服务质量。在司法领域中可供挖掘的一大部分数据为文本格式，包括诉讼、纠纷中所保留的裁判文书、案情描述文件等。因此，在司法领域中，对文本数据的自然语言分析与挖掘具有较为重要及深远的意义，具备研究的条件与价值。

对自然语言处理方向的深度学习系统性能的提升，就是其中一个值得研究的问题。能够实现落地的深度学习系统都应该具备较高的预测准确率以及对输入数据的较强的鲁棒性。而当前司法领域中的自然语言处理系统的性能还不够完善，具体表现为预测系统准确率低、缺乏对模型的多维度测评等。

本文的主要研究内容为：首先本文针对司法领域深度学习系统数据缺乏的问题，提出并设计了司法文本数据生成框架，分为训练数据与测试数据生成两大功能模块，统一了司法文本生成流程以及数据处理规范等。其次，本文调研与分析了现有的文本数据扩增方法与深度学习模型多维度测试指标，结合司法文本特点与司法行业特性，设计并实现了司法文本训练数据与测试数据生成算法，并对算法的可用性做出实验论证。

本文的主要工作分为两大部分，一部分为司法文本数据生成算法的设计与实现。论文参考现有的通用数据扩增方法，并根据司法文本数据的特点，设计符合司法文本特点的数据生成方式。对于训练数据生成，本文设计了基于规则的生成算法，对司法数据中固有的词组进行标记与修改以生成相似的文本；设计了基于变分编码器的生成算法，学习文本的数据映射关系，生成与原始数据分布相似的新数据。对于测试数据生成，本文设计了含噪声项数据生成算法，以解决对模型抗噪能力的测试问题；设计了基于遗传算法的对抗攻击测试数据生成算法，以解决对模型抗对抗攻击能力的测试问题。

另一部分为数据生成 web 系统的设计与实现。本文拟设计一套便于用户交互的司法文本数据生成 web 系统，主要由前端 UI 及展示层、数据生成层、数据处理层与数据存储层组成。前端 UI 及展示层主要负责为用户提供系统使用界面。通过 HTTPS 协议将用户请求及文件从客户端发送至服务器端，并在服务器处理请求后将结果发回至客户端展示。数据生成层为系统的核心业务层，负责提供不同形式的生成数据文件，从数据处理层获取用户上传的经过处理的格式化文本数据，根据用户自定义参数调整生成方式，最终为用户提供生成的文本文件。数据生成层分为训练数据生成、测试数据生成两大基本模块。数据处理层负责读入用户传输至系统的文本文件，并进行文本进行格式化处理，以及数据预处理等流程，将经过处理的文本交付至数据生成层。数据存储层由 MySQL

数据库与 HDFS 文件管理系统组成。其中，MySQL 数据库负责存储结构化信息，包括用户信息（用户名、密码、用户权限等）及文件信息（文件名、文件实际存储名称、文件所属用户等）。HDFS 文件管理系统负责存储用户上传及生成的文本文件及模型文件。

1.4　本文的组织结构

本文将提供设计完善的司法领域文本数据的生成算法，并将各生成方法进行统一设计与封装，以 web 应用作为系统载体，以便于用户交互。本文的组织结构如下：

第一章引言，介绍了项目背景及意义、国内外在该方向的研究现状以及本文的主要研究工作。

第二章技术综述，将叙述文本所涉及的相关技术，包括自然语言处理相关技术、变分编码器、Django 框架技术、分布式存储技术以及遗传算法等。

第三章系统需求分析与架构设计，结合应用场景对系统提出功能性及非功能性需求，对系统的整体架构进行了设计。并根据功能对系统进行模块划分，对各个模块分别进行了详细的设计。

第四章系统实现，以系统需求分析及架构设计为基础，叙述了系统各个模块的详细实现过程，并对系统功能进行了实验及案例分析，最后对系统进行了测试。

第五章总结与展望，将总结系统所做出的工作，提出系统存在的问题，对系统在功能拓展及推广方面的改进方向进行了展望。

第二章　技术综述

2.1　自然语言处理技术

本文系统功能为对司法文本数据进行生成，在生成中应用到了对中文文本的处理操作。包括对中文词语的分割、利用词向量使词语间语义相关性得到体现、利用语言模型预测句子相邻词语出现的概率等。本文介绍了所用到的自然语言处理技术背景，包括中文分词技术、词向量和语言模型。

2.1.1　中文分词技术

中文字词结构的性质与英文不同，英文的词语间具有空格作为词语的分隔符号，而中文的词语并不存在天然的分隔符号，因此在进行中文自然语言处理过程中，需要将分词作为最基本的操作步骤，分词的效果直接影响了是否可以匹配到正确语义的词语。中文分词方法根据实现原理和特点分类，主要分为基于词典匹配的方法与基于统计的机器学习方法 [20]。

基于词典的方法的原理是字符串的匹配 [21]，该方法基于高质量的词典进行词语的规则匹配，常见的基于词典的分词算法包括正向最大匹配法、逆向最大匹配法、双向匹配分词法等。该方法的优点包括分词速度快、领域适应性强等，缺点为对于句子语境不敏感，具有分词的歧义性、以及对于词典中部分词语有缺失问题时无法分词的问题。

目前应用较为广泛的方法是基于统计的分词方法。基于统计的分词方法常用算法包括隐马尔可夫链模型（HiddenMarkovModel，HMM）[22]、条件随机场模型（CRF）[23]、支持向量机（SVM）[24] 等机器学习算法。也存在基于深度学习的分词模型，例如使用双向长短期记忆网络与条件随机场结合的模型。基于统计的分词方法的基本思路为对句子进行词语的标注训练，其优点为学习能力强，对分词的歧义性解决能力强，缺点为领域适应性弱。

现有的中文分词工具采用的分词策略为基于统计的分词方法，同时结合基于词典的方法，包括大数据搜索与挖掘共享平台（NLPIR）、THULAC（THU Lexical Analyzer for Chinese）[25]、结巴分词器（jieba）、HanLP 分词器等。

系统采用了 jieba 分词器，其结合了基于规则与基于统计的方法，对于词典中存在的词，算法基于前缀词典进行扫描，形成层级包含结构，使用动态规划找出最大概率路径；对于不存在词典中的词，算法将词组按位置作为标注状态，构造一个经过分词的训练数据集，利用 HMM 模型，得到各个参数，采用 Viterbi

算法进行推导，得到分词结果。

在本文中关于 jieba 分词器的应用不仅包括分词，也包括文档关键词的提取。jieba 分词器利用 TF-IDF 算法 [26] 计算经过分词的词语在文档集中的重要程度。TF-IDF 算法由两部分 TF（词频）与 IDF（逆文档频率）组成，TF-IDF 的值为 TF 与 IDF 的乘积，其含义为，如果一个词在其文档上出现的频率越高，且包含该词的文档在文档集中越少，说明该词与该文章高度相关。TF 表示某个词语 w 在文挡中出现的频率，也就是说，该词语出现的次数 t_w 占所有词语数量的比例。IDF 表示包含词语 w 的文档数量的占比，包含词语 w 的文档数量越少，词语越重要。如公式 2.1， 2.2所示。

$$TF_w = \frac{t_w}{\sum_{i=0}^{N} t_i} \tag{2.1}$$

$$IDF_w = \log \frac{\sum_{i=0}^{M} d_i}{f_w(\sum_{i=0}^{M} d_i) + 1} \tag{2.2}$$

2.1.2　词向量技术

在自然语言处理过程中，在对文本进行分词、停用词去除等格式化操作后，要进行词语的向量化表示。词语的表示方法包括离散表示（one-hot representation）与分布式表示（distribution representation）。

离散表示以词表大小作为向量长度，在向量中将该词在词表的位置处的值置 1, 其余位置均为 0。例如：

计算机：[0,0,0,0,0,0,1,0,0,0,.....,0]

草莓：[0,0,1,0,0,0,0,0,0,0,.....,0]

离散表示存在的问题之一为向量长度太长，且向量过于稀疏，使得特征空间过于庞大；问题之二为缺乏词与词间的语义相关性，语义相近的词语向量间无任何联系，导致后续模型的泛化能力弱。

分布式表示将词语表示为定长的、连续的、稠密的向量。词向量（word embedding）就是一种分布式表示的常用方法。词向量的生成包括基于统计的生成方式与基于语言模型的生成方式。

本文所使用词向量的生成方式为基于语言模型的生成方式。基于语言模型的生成方式通过训练语言模型，对出现在上下文环境中的词语进行预测，在训

练过程中模型中间的隐层向量即为词向量。生成词向量的方法包括 CBOW [27]、Skip-gram、FastText、GloVe [28] 等。本文所使用的词向量为 FastText 的中文预训练词向量 [29]。其训练结构如图 2.1所示。

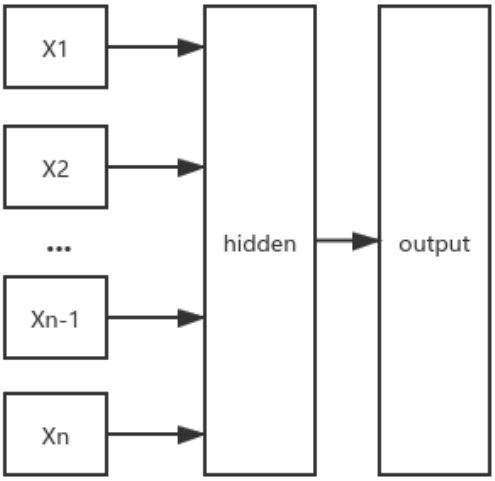


图 2.1: FastText 词向量模型结构图

FastText 模型包含三个层级结构：输入层（input）、隐藏层（hidden）、输出层（output），其模型结构与 word2vec 词向量中的 CBOW 模型结构相同。FastText 模型在训练时以词语的序列作为输入，输出该序列属于不同类别的概率。在训练中，输入的词和 N-gram 词组组成特征向量 x_i，输出经过非线性激活函数 softmax 预测为分类标签。经过在大量语料上的训练，其隐藏层向量即可作为预训练词向量使用。

N-gram 特征是语言模型中提出的概念，原理是在词语的序列中以 n 为窗口大小形成词组。FastText 在训练中使用 N-gram 词组可以学习到词语的上下文信息，从而在生成词向量时可以学习到单词的局部词序信息。

由于 FastText 模型层级简单，且对 softmax 做了层级优化，改善了其运算效率，因此很适合大量语料库的高效训练。

2.1.3　语言模型

语言模型的功能为：对于语言序列 $w_1, w_2, w_3, ..., w_n$，计算语言序列出现的概率，即 $P(w_1, w_2, w_3, ..., w_n)$，以判断该语言序列是否为合乎语法逻辑的正常语句。常用的语言模型分为两种类型：统计语言模型与基于神经网络的语言模型。

常用的统计语言模型为 N 元文法模型（N-gram Model），该模型将词序看作

随机事件，描述某种特定词序的出现概率。由链式法则可以得出，衡量由 n 个词组成的句子 $S = (w_1, w_2, w_3, ..., w_n)$ 的概率公式为：

$$P(S) = p(w_1 w_2 ... w_n) = p(w_1)p(w_2|w_1)...p(w_n|w_{n-1}...w_2 w_1) \tag{2.3}$$

对于任意长度的语句，利用极大似然估计计算 $p(w_i|w_{i-n+1}...w_{i-1})$ 不现实，因此引入马尔可夫假设，假设某个词出现的概率只依赖于该词的前 $n-1$ 个词。根据此假设定义了 N-gram 语言模型：

n=2 bigram: $P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i|w_{i-1})$

n=3 trigram: $P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i|w_{i-2}, w_{i-1})$

统计语言模型直观、可解释性强，但其算法仅基于统计频次，泛化能力弱。另一方面，当 n 较小时，无法获得上文的长时依赖；当 n 过大时，许多预测概率结果为 0，导致稀疏问题。

Y. Bengio 等人首次提出神经网络语言模型 (neural network language model, NNLM) 的概念 [30]。基于神经网络的语言模型提出了词向量的概念，与统计语言模型使用词语的离散表示相较，基于神经网络的语言模型采用分布式的表示方法，降低了词向量的维度，并利用词语间相似性提高了模型泛化能力。

NNLM 由于其网络结构较简单，仅由输入层、嵌入层、隐藏层与输出层组成，因此对于较长词序列的预测不准确。为了解决较长词序列的长期依赖问题，RNNLM [31] 利用 RNN 网络训练语言模型，由于 RNN 网络在时间序列上的记忆优势，RNNLM 可以捕获更长的依赖。

ELMo 模型 [32] 采用双向 LSTM 网络结构，由前向语言模型与后向语言模型组成，前向语言模型用于获取上文信息，后向语言模型用于获取下文信息，因此可以双向预测词语的出现概率。其训练过程中的目标函数为前向与后向语言模型的最大似然。但双向 LSTM 属于弱双向性结构，因此获得的不是真正意义上的上下文信息。

BERT 模型 [33] 利用 Transformer 编码器作为模型结构，属于 self-attention 机制，模型上下层直接相连，属于真正意义上的双向的语言模型。在 BERT 模型的预训练过程中有两项预测任务，其一为随机将句子中部分词语隐藏，将隐藏词语输出的向量输入 softmax，来预测该词语；其二为句子级别的预测任务，预测输入的两段文本是否属于连续的文本。

2.2　变分编码器

生成网络模型是无监督方式下学习数据分布的方法。其作用为学习训练集的真实数据分布，以便生成与原有数据具有一些变化的、新的数据点。

自动编码器（Auto-encoder）用于将输入的图像或文本编码为低维向量，也就是利用较小空间的表现形式存储数据分布的潜在信息。使用相应的解码器可以将编码器生成的低维向量进行解码，映射为输入表现形式。自动编码器在解码后只能形成原有的表现形式，无法生成与输入相似的表现。

变分编码器（Variational Auto-encoder，VAE）[34] 是近年来流行的无监督的、学习复杂数据分布（例如图像、文本等数据）的生成网络模型之一。VAE 为基于贝叶斯推断的概率模型，即该模型的目标为学习训练数据的潜在概率分布，以便从该概率分布中采样新数据，新数据与原数据具有相似的概率分布，因此其表现相似。

其原理为：对于待模仿的概率分布 $P(z)$，当任意给定一种概率分布 $Q(z)$，总存在一个可测函数，将概率分布 $Q(z)$ 映射到待模仿分布 $P(z)$，使 $Q(z), P(z)$ 分布相近。VAE 旨在学习训练数据的低维潜在表现形式，这些潜在变量可以存储模型需要生成的输出类型的有用信息。VAE 使用容易找到的，更简单的分布 $Q(z|X)$ 对 $P(z|X)$ 进行建模，其中 X 为输入的训练数据。并使用 KL 散度最小化 $Q(z|X)$ 与 $P(z|X)$ 之间的差异。如公式 2.4所示：

$$\log P(X) - D_{KL}[Q(z|X)||P(z|X)] = E[\log P(X|z)] - D_{KL}[Q(z|X)||P(z)] \tag{2.4}$$

其中，$Q(z|X)$ 为编码器网络，z 为对数据 $x(x \in X)$ 的编码表示，而 $P(X|z)$ 为解码器网络。

VAE 的目的是在给定 $D_{KL}[Q(z|X)||P(z|X)]$ 某些误差的条件下，使数据分布的对数似然最大。VAE 希望潜在表示向量 z 接近高斯分布，因此假设 $P(z)$ 为 $N(0,1)$，按照此假设，$Q(z|X)$ 也接近该分布。假设 $Q(z|X)$ 其均值为 $\mu(X)$，标准差为 $\sigma(X)$，得到解决方案如公式 2.5所示。

$$D_{KL}[N(\mu(X), \sigma(X))||N(0,1)] = \frac{1}{2}\sum_{k}\Big(\exp(\sigma(X)) + \mu^2(X) - 1 - \sigma(X)\Big) \tag{2.5}$$

因此，VAE 的损失函数包含两项：其一为输入到输出的重建损失，其二为上述公式中的 KL 散度。因此可以使用反向传播算法对 VAE 网络进行训练。由于梯度不可以随随机节点计算，因此，VAE 网络采用重构参数化的技巧，使分布 $N(\mu(X), \sigma(X))$ 可以随机传播至 z。首先从分布 $N(0,1)$ 采样参数 ε，然后通过 $\mu(X) + \varepsilon \cdot \exp(\sigma(X)/2)$ 计算获取 z 值。

VAE 的训练过程示意图如图 2.2所示。训练的过程即是对输入样本 X 编解码过程，Q 将输入样本编码为低维表示 z，P 将低维表示 z 解码为 $f(z)$，使重构误差尽可能小。

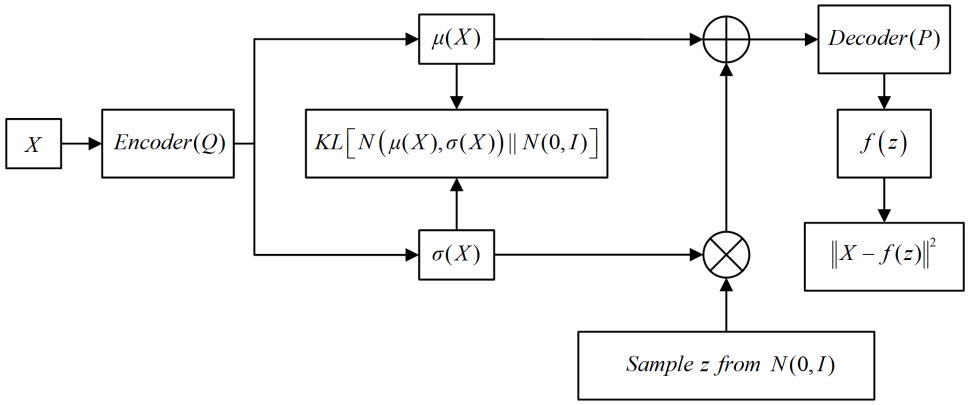


图 2.2: VAE 训练过程示意图

2.3　Django 框架

本文考虑到系统对于用户的易用性和可交互性，使用 web 应用程序开发 B/S 系统架构，用户可通过浏览器访问系统服务。Django [35] 是基于 Python 语言的开源 web 框架，具有易维护、易管理、可拓展的优点。Django 框架中解决了 web 应用中大部分的配置与网络交互问题，因此用户可以专注于编写应用程序本身而无需关注底层实现。

在 web 应用程序中，大部分开发框架所使用的设计模式为 MVC 模式（Model、View、Controller）。web 开发框架将三者以松耦合的方式连接，三者分别负责与数据库的连接、与用户的交互以及将用户请求应用于业务逻辑。Django 属于 MTV 模式，其原理与 MVC 相似，其定义的功能如表 2.1所示。

表 2.1: Django 的 MTV 模式功能

层次	职责
模型（Model）	负责与数据库的连接与交互，包括数据库表的定义、数据的存取、数据的验证等操作
模板（Template）	负责与页面的展示及与用户的交互，定义页面的展示模板，及业务逻辑写入页面模板等操作
视图（View）	负责业务逻辑的调用，并在业务逻辑中调用模型与模板中的内容

Django 中用户通过浏览器访问应用时返回一个 web 请求的工作流程如下：

用户通过浏览器访问服务页面url,web服务器将请求发送至WSGI。WSGI(Web Service Gateway Interface)，即 web 服务器网管接口。它定义了 web 服务器与 python web 应用间的接口标准，以保证 web 服务器可以与 python web 应用互相通信。服务器将用户请求的 http 报文转换为 WSGI 的规定格式，继而传递给 python 应用。符合 WSGI 标准的 http 处理函数接受两个参数，参数一格式为字典，包含请求的信息，如请求方式、请求路径等；参数二为在可调用对象中调用的函数，用于发起响应，包含状态码、headers 等。

URLConf 用于为用户访问的 url 寻找 View，并在 view.py 文件中被定位至相应的业务逻辑函数。Django 框架的 url 映射器通常存储在 urls.py 文件中，url 映射器定义了路由与相对应的视图间的映射列表，当收到的 HTTP 请求具有与模式匹配的 url，按定义顺序逐个匹配 urlconfig 的正则部分，匹配成功将调用其关联的视图函数并将 request 传递到该函数。

视图是 web 应用的核心，负责处理请求的函数，接受 HTTP 请求，并返回 HTTP 响应。视图通过 Model 访问满足要求所需的数据，在经过业务逻辑函数处理后，得到格式为 Python 字典对象的 HTTPResponse 对象，传递给 templates。Model 负责处理 web 应用程序的管理，以及数据库中数据的查询。models 定义了存储数据的结构，包括字段类型、默认值等。Model 的定义独立于基础数据库，Django 提供了用于搜索数据库的查询 API，在视图函数中无需直接与数据库交互。

Response 对象被送入对应的模板文件，填写模板文件中的变量，渲染为用户可交互的 web 页面。Django 框架在模板渲染时，创建 Template 对象，利用 context 从 views 中的逻辑函数传递数据，Context 类的构造函数由字典映射变量与其对应的值组成，Template 对象调用 render 方法，将 context 传递到 html 文件中。在 html 文件中，python 模板语言的语法使用“{%%}”包裹，而数值传递利用“{{}}”包裹。

2.4　遗传算法

遗传算法（Genetic Algorithm）[36] 是模拟生物学自然进化理念提出的一种启发式优化算法。该算法的生物学原理为：在种群的每一代个体中选择最适合生存的个体作为父本进行繁殖，产生下一代。下一代个体继承了父代的特征，经过基因重组，即结合父本与母本的特征，类似 DNA 的交叉过程，经过变异过程，即随机产生父本母本不存在的特征，最终产生子代。子代可能产生比父代更适合生存的个体，这个过程不断重复，种群不断繁殖，最终找到最适合生存的个体。将遗传算法应用于优化问题的工作流程如下所示，流程图如图 2.3所示。

1. 随机初始化第一代种群，种群中的个体为待优化问题的随机解。

2. 判断种群中的个体是否存在符合标准的个体，如果符合，输出最佳个体并结束算法；否则进行下一步。

3. 在种群中随机选择个体生成下一代。选择概率与种群中个体在适应性函数 (fitness function) 上的表现成正比。适应性函数用于判断种群内个体的优劣，更优的个体拥有更高的适应性函数评分，表示其具有更强的适应度。

4. 利用在 3 中选择的样本进行下一代的生成。生成主要分为两部分，第一步是交叉（crossover）过程。将父代个体两两分为一组，相互交换表现特征，形成两个新的个体。第二步是变异（mutation）过程。将交叉生成的新个体的部分特征进行随机替换，替换为其父母不具有的特征值，形成新个体，也就是子代个体。其功能为在子代中引入不同于父代的新特征，防止陷入局部最优解。

5. 将经过交叉与变异的子代种群返回步骤 2 进行判定，直到在设置的迭代次数中产生最优解。

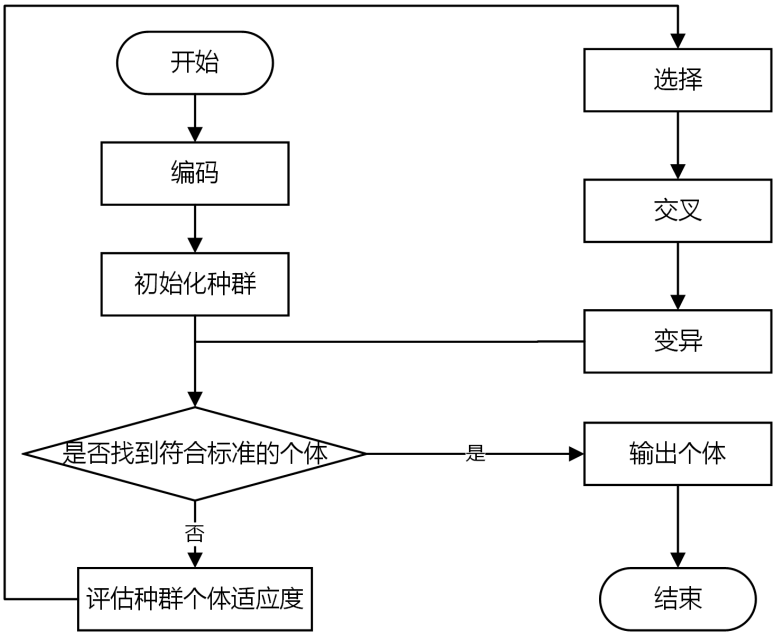


图 2.3: 遗传算法流程图

2.5　分布式存储技术

在大数据场景下，大量数据及文件的存储对服务器的性能提出了更高的挑战。传统的集中式数据管理架构存在许多问题，存储空间有限且不可拓展、存在单点故障问题等。分布式存储技术管理架构的原理为利用廉价、可拓展的计算机集群，将服务器数据分布式地存放在多个计算机，也就是分布式节点上。这样有利于解决存储空间的瓶颈问题，提高了可拓展性。但分布式存储技术的实现需要解决数据一致性、可用性、分区容错性等问题，实现较为复杂。

Hadoop [37] 是由 Apache 开发的分布式系统架构。是在分布式的集群中存储大量数据，并在分布式集群中进行计算的实现。该架构使用简单的程序模型跨计算机对大型数据进行分布式的处理，由于集群可能出现故障，因此 Hadoop 架构设计了检测和处理故障的机制，使集群可以提供高可用性的服务。

HDFS，即 Hadoop 分布式文件系统（Hadoop Distributed File System）[38] 属于 Hadoop 中的核心组件之一，为 Hadoop 提供底层的存储系统。HDFS 支持文件数据的分布式存储，将文件分布存储在多台服务器上。其副本存放策略、心跳检测机制及块报告机制等使其具有高容错、高可靠性，流式访问模式使其适合批处理及大数据处理，为较大数据集的存取提供了安全、高速的支持。

HDFS 主要由四部分组成：HDFS client、NameNode、DataNode、Secondary NameNode。HDFS client 为访问 HDFS 的客户端，利用命令与 HDFS 交互，提供启动、关闭、访问与管理等服务。图 2.4为 HDFS 模块关系图。

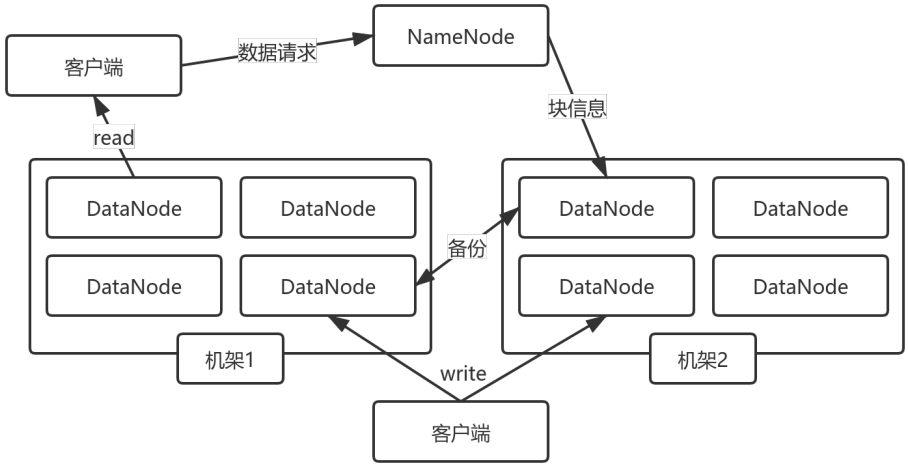


图 2.4: HDFS 模块关系图

NameNode 用于与 client 的读写请求交互，为 client 提供文件位置信息，并

承担对 Block、副本策略等的管理功能。NameNode 与数据节点采用松耦合的方式连接，集群可以增加或减少节点，来动态适应服务器容量的实时需求。

DataNode 与 NameNode 持续通信，查看是否有需要执行的任务。持续的通信可确保 Namenode 对每个 DataNode 保持控制。负责执行 NameNode 提供的读写命令，并负责将文件分配存储至 Block 及从多个 Block 读取文件数据。当某个 DataNode 节点无法正常工作，NameNode 将立即将该节点的任务重新分配到包含相同数据块的另一个节点。

Secondary NameNode 辅助 NameNode 的一部分管理工作，负责合并文件信息并传送至 NameNode。Block 为 HDFS 存储文件的单位，在 Hadoop2.0 中 Block 大小为 128MB。

2.6　本章小结

本章首先介绍了本文所涉及的一些自然语言处理技术，包括中文分词技术、词向量技术及语言模型，为接下来的文本处理打下了理论基础；然后介绍了变分编码器及其在文本数据生成中的应用，明确了基于变分编码器的数据扩增手段的技术框架；介绍了遗传算法的原理与基本结构，为文本基于遗传算法的对抗攻击技术的实现提供了理论依据；最后介绍了本文的系统架构的框架工具，及本文为支持大数据存储与计算所应用的技术工具。

第三章　系统需求分析与架构设计

本章基于司法深度学习模型在开发过程中对于文本数据的真实需求，转化为司法文本数据自动化生成系统的产品需求。基于需求分析的结果，对系统进行了整体的架构设计。进一步，将系统按照功能划分为四个子模块：数据交互模块、文本预处理模块、数据生成模块以及用户文件管理模块，对各个模块分别进行了详细的功能设计，以及对系统数据库进行了设计。

3.1　系统需求分析

作为完整的软件系统，系统满足用户管理功能，包括用户注册系统、登录系统、管理员管理用户信息等基本功能。本文主要对数据交互、文本预处理、数据生成以及用户文件管理四部分需求内容进行分析，系统用例图如图 3.1所示。

数据交互功能负责用户文件的上传与下载，包括用户待生成文件的上传、预测模型的上传、用户已生成文件的展示、用户已生成文件的下载；负责与 HDFS 文件系统建立数据交互接口，实现文件在集群中的分布式存储。

文本预处理功能负责对用户上传的文本文件进行解析。包括输入文件的结构化操作、文本的分词操作、以及文本对应语料库字典的映射操作等。

数据生成功能是系统的核心功能，负责利用系统提供的多种数据生成方式，生成不同形式的新文本。包括训练数据生成与测试数据生成两种生成子功能。

用户文件管理功能负责对系统用户生成的文件进行管理，包括历史生成文件的展示、文件下载及文件删除。系统将功能低耦合地封装在几个模块中，各模块彼此调用，具有较强的可扩展性与可修改性，方便系统的迭代修改。

各功能模块的调用顺序与关系大致为：首先，用户登录系统，进入系统主页面，选择生成方式。系统将调用数据交互功能模块，上传生成文本所需的文本文件或模型文件。上传后，系统调用文本预处理功能模块，对用户输入的非结构化文本进行结构化处理，以及分词、词典映射等用于支持数据生成的预处理步骤。预处理完成后，调用系统核心模块：数据生成模块，根据用户选择的生成方式及参数，传入经过预处理的文本文件，进行文本数据的生成。最后，系统再次调用数据交互功能模块，用于将生成文件下载至用户本地。

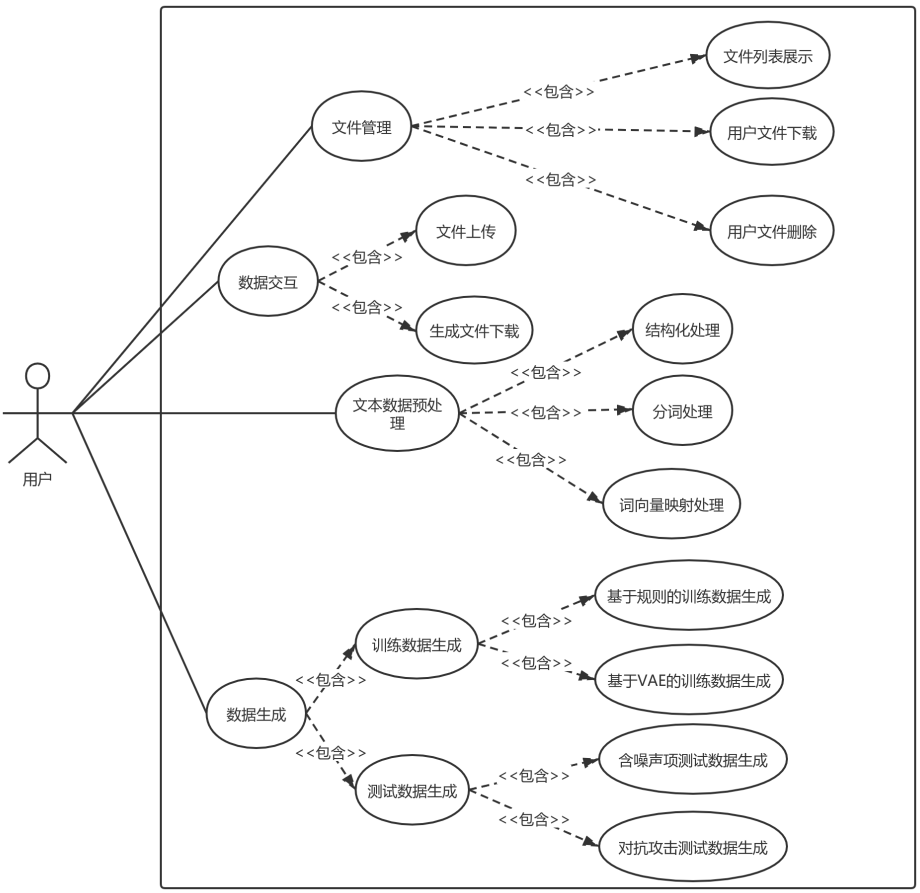


图 3.1: 司法文本数据自动化生成系统用例图

3.1.1　数据交互需求分析

数据交互功能是数据生成的基础，用户需使用数据交互功能上传待生成的文本文件，才能进行后续的生成步骤，数据交互功能负责文件的上传与下载。表 3.1提供了数据交互的用例描述。

数据交互功能与 HDFS 文件系统连接，需要将用户上传的文件存储进 HDFS 文件系统，以提高文件存储的可拓展性。当用户下载文件时，需要将 HDFS 中的文件下载至用户本地计算机。在存储时为防止相同文件重复存储，系统需要计算文件 sha1 摘要作为文件存储的文件名。文件相关信息（用户文件名、存储文件名、文件大小、上传时间）则存储在结构化的 MySQL 数据库中，以方便文件的查找。在文件下载时，获取数据库中的存储文件名后在文件系统中查找文件。

表 3.1: 数据交互用例状态描述表

ID	UC1
名称	数据交互
描述	用户与系统间进行文件数据交互，支持文件的上传与下载
参与者	普通用户
前置条件	用户登录系统；下载文件需在生成之后
后置条件	无
优先级	高
正常流程	1. 用户根据系统对文件的格式要求，上传所需文件 2. 系统读取上传的文本文件，并计算其 sha1 哈希值，作为 HDFS 文件系统内存储的文件名 3. 上传文件至 HDFS 文件系统 4. 写入文件信息至 MySQL 数据库，包括用户文件名、存储文件名、上传用户、文件大小、文件上传时间 5. 在数据生成完成后，用户点击下载生成的文本文件 6. 系统根据其文件名在 HDFS 文件系统中查找并下载至用户本地
异常流程	1a. 若用户上传的文件格式与要求不符，则返回提示信息，重新上传 3a. 若系统内存在相同文件名，则不再上传

3.1.2　文本预处理需求分析

表 3.2提供了文本数据预处理的用例描述。文本预处理模块负责在上传输入文件后，对输入的文本文件进行预处理操作，包括：1）在用户上传文件后，首先进行结构化操作，将文本文件结构化为字典格式的 json 文件，方便后续格式的添加；2）进行分词操作，调用分词 API，将文本按词分割，支持后续的数据生成；3）当用户选择基于 VAE 的训练数据生成方式或对抗攻击测试数据生成时，进行词典映射，将文本文件中的词语映射为语料库词典序号。文本预处理模块将结构化的 json 文件替代用户的非结构化输入文件，存储在 HDFS 文件系统中，以便支持后续的数据生成操作。

3.1.3　数据生成需求分析

数据生成功能是系统的核心功能，可以为用户提供全面的司法文本数据生成服务，分为训练数据生成及测试数据生成两部分。训练数据生成包括基于规则的数据生成方式与基于 VAE 的数据生成方式。测试数据生成包括含噪声测试数据的生成与对抗攻击测试数据的生成。

表 3.3提供了基于规则的训练数据生成的用例描述。基于规则的训练数据生成方式根据法律文本的特征所设计，利用规则匹配寻找法律相关的专业术语或关键词语，对这部分词语进行操作生成文本数据。基于规则的训练数据生成包括 Blind 扩增、反事实扩增、同义词替换三种具体规则。支持用户调节基于规则

表 3.2: 文本预处理用例状态描述表

ID	UC2
名称	文本预处理
描述	将上传的文本文件做预处理以支持后续生成操作
参与者	普通用户
前置条件	用户上传文本文件
后置条件	返回生成的文本文件结果，用于进行数据生成
优先级	中
正常流程	1. 用户上传文件后，从 HDFS 文件系统中加载文件并进行结构化操作，将文件内容、标签等以 json 形式保存 2. 结构化操作完成后，调用分词 API，对文件中的句子进行分词操作，并存储在 json 文件中 3. 当用户选择基于 VAE 的训练数据生成方式或对抗攻击测试数据生成时，将词语映射为词典序号，将映射结果存储在 json 文件中 4. 将结构化的 json 文件存储在 HDFS 文件系统
异常流程	3a. 对于未在词典内出现过的词，标记为“unknow”

的训练数据的生成参数，使数据生成的结果定制化。

表 3.3: 基于规则的训练数据生成用例状态描述表

ID	UC3
名称	基于规则的训练数据生成
描述	对用户上传的文本文件进行基于规则的数据生成，生成与原数据分布相同的新数据
参与者	普通用户
前置条件	用户上传的文本文件经过预处理
后置条件	返回生成的文本文件至数据交互模块
优先级	高
正常流程	1. 用户选择生成规则并调整各项生成参数 2. 系统从 HDFS 文件系统中取出经过文本预处理的文件 3. 系统传入用户参数及文本预处理文件，调用基于规则训练数据生成函数 4. 生成完成后存储文件至 HDFS 文件系统，并将文件信息存储至 MySQL 数据库中 5. 返回生成的文本文件信息至数据交互模块
异常流程	无

表 3.4提供了基于 VAE 的训练数据生成用例描述。变分编码器（VAE）是一种无监督的学习数据分布的机器学习模型，基于 VAE 的训练数据生成方法在用户输入训练数据与测试数据文件及调整训练超参数后，系统训练 VAE 生成模型，学习输入文本的数据分布。在生成模型训练结束后，模型可根据用户输入的生成数量，生成与原始文本分布相近的新文本。由于 VAE 模型的训练耗时较长，因此在训练结束后，系统将在文件系统中保存 VAE 模型。当用户进行数据生成时，调用经过训练的模型，可支持用户的多次生成。

表 3.4: 基于 VAE 的训练数据生成用例状态描述表

ID	UC4
名称	基于 VAE 的训练数据生成
描述	根据用户上传的文本文件，训练 VAE 生成模型，生成与原数据分布相同的新数据
参与者	普通用户
前置条件	用户上传的文本文件经过预处理
后置条件	返回生成的文本文件至数据交互模块
优先级	高
正常流程	1. 用户调整训练 VAE 生成模型的超参数 2. 系统从 HDFS 文件系统中取出经过文本预处理的 json 文件 3. 系统传入用户参数及文本预处理文件，训练 VAE 生成模型 4. 用户选择训练完成的 VAE 生成模型，可查看训练详情 5. 用户选择生成文本数量，系统利用该生成模型生成文本 6. 生成完成后存储文件至 HDFS 文件系统，并将文件信息存储至 MySQL 数据库中 7. 返回生成的文本文件信息至数据交互模块
异常流程	无

表 3.5提供了含噪声项测试数据生成的用例描述。含噪声项测试数据生成根据系统定义的规则改变原始数据的分布，使生成的新数据具有某种不同于原始数据的分布特性，可用于作为测试数据来测试模型在抗噪能力方面的性质。含噪声项测试数据生成包含三种噪声项的生成方式：含乱序项测试数据生成、含缺失项测试数据生成、含冗余项测试数据生成，分别对应于三种不同的含噪声项数据分布特性。支持用户调节含噪声项测试数据的生成参数，使数据生成的结果定制化。

表 3.5: 含噪声项测试数据生成用例状态描述表

ID	UC5
名称	含噪声项测试数据生成
描述	生成具有噪声项的，与数据原分布不同的测试文本数据
参与者	普通用户
前置条件	用户上传的文本文件经过预处理
后置条件	返回生成的文本文件至数据交互模块
优先级	高
正常流程	1. 用户选择需要生成的噪声选项，调整各项生成参数 3. 系统从 HDFS 文件系统中取出经过文本预处理的文件 4. 系统传入用户参数及文本预处理文件，调用含噪声项生成函数 5. 生成完成后存储文件至 HDFS 文件系统，并将文件信息存储至 MySQL 数据库中 6. 返回生成的文本文件信息至数据交互模块
异常流程	无

表 3.6提供了对抗攻击测试数据生成的用例描述。对抗攻击测试数据生成要

求用户输入需要攻击的预测模型与测试数据，系统将基于遗传算法对原始测试数据进行修改，使修改后的测试数据的预测标签与修改前相比发生改变。系统将为用户提供对抗攻击成功的测试数据，并统计在测试数据上的攻击成功率与平均修改比例。

表 3.6: 对抗攻击测试数据生成用例状态描述表

ID	UC6
名称	对抗攻击测试数据生成
描述	对用户上传的预测模型进行对抗攻击，对测试数据进行修改，生成可以攻击模型成功的测试数据
参与者	普通用户
前置条件	用户上传的文本文件经过预处理
后置条件	返回生成的文本文件至数据交互模块
优先级	高
正常流程	1. 用户调整对抗攻击的各项参数 2. 系统从 HDFS 文件系统中取出经过文本预处理的文件以及对抗的预测模型 3. 系统传入用户参数及文本预处理文件、预测模型，调用对抗攻击函数 4. 生成完成后存储文件至 HDFS 文件系统，并将文件信息存储至 MySQL 数据库中 5. 返回生成的文本文件信息至数据交互模块
异常流程	无

3.1.4　用户文件管理需求分析

表 3.7提供了用户文件管理的用例描述。用户的文件列表可查看历史生成过的所有文件信息，包括文件名、文件大小、文件上传时间、文件生成方式等。并支持用户多次下载文件至本地。系统中设置了定义文件重复次数的数据类型，当用户删除文件时，若此文件为重复文件，则删除该文件在数据表中的定义，若此文件非重复文件，则在 HFDS 文件系统中删除该文件。

3.1.5　非功能性需求分析

非功能性需求分析可以评估系统的健壮性与鲁棒性，对软件系统的作用同样不可或缺。本节从安全性、可靠性、容错性三个方面对系统的非功能性需求进行叙述。

安全性。用户的文件只允许被本人或系统管理员访问与操作，本系统的安全性要求有：用户只能看到自己的生成文件；只有管理员可访问系统内所有文件及用户信息。

可靠性。系统在用户提交不合理操作时，应返回合理提示，提示用户重新操作，而不是造成系统崩溃或无法继续操作。在本系统中的安全性要求包括：用户传入的文件格式不合理时应提示用户重新上传；用户信息填写错误应提示其重

表 3.7: 用户文件管理用例状态描述表

ID	UC7
名称	用户文件管理
描述	用户查看历史生成的文件列表并管理文件
参与者	普通用户
前置条件	用户登录系统
后置条件	无
优先级	中
正常流程	1. 用户点击“个人中心”按钮，进入文件列表 2. 显示用户历史生成过的文件信息列表 3. 用户点击“下载”按钮，浏览器将下载该文件至本地；点击“删除”按钮，删除服务器中的文件及数据库信息
异常流程	2a. 若用户未生成过文件，则显示空列表

新填写等。

容错性。系统中的用户上传或生成的文件应在分布式集群中使用冗余手段备份，不会因某一台设备故障导致用户文件的丢失。

3.2　系统架构设计

司法文本数据自动化生成系统的总体结构如图 3.2所示，包括前端 UI 及展示层、数据生成层、数据处理层和数据存储层。本节对平台进行了总体设计，确定了系统框架。

前端 UI 及展示层主要负责为用户提供系统使用界面。通过 Bootstrap 前端框架及 jQuery 框架搭建了 web 前端界面，借助浏览器与用户交互，方便用户操作。通过 Django 框架中的模板引擎渲染功能，将业务逻辑数据写入前端页面模板中。通过 HTTPS 协议将用户请求及文件从客户端发送至服务器端，并在服务器处理请求后将结果发回至客户端展示。

数据生成层为系统的核心业务层，负责提供不同功能下的数据生成计算服务。数据生成层从数据处理层获取用户上传的经过处理的格式化文本数据及其他所需文件，并根据传入的用户自定义生成参数，进行数据生成计算。数据生成层分为训练数据生成与测试数据生成两大基本模块。两大模块下均提供了不同生成方式，包括基于规则的训练数据扩增、基于变分编码器的训练数据生成、含噪声项的测试数据生成、对抗攻击测试数据生成。

数据处理层介于数据存储层与数据生成层之间，负责输入文本文件的预处理服务。本层的具体职责有三：一是读入用户传输至系统的文件，并进行格式化处理；二是对文本文件进行分词处理，为一下步数据的生成做准备；三是根据参

数将文本文件中的词语替换为语料库中对应的词典序号，为基于变分编码器的训练数据生成及测试抗对抗攻击能力的测试数据生成模块做数据准备。

数据存储层由 MySQL 数据库与 HDFS 文件管理系统组成。其中，MySQL 数据库负责存储结构化信息，包括用户信息（用户名、密码、用户权限等）及文件信息（文件名、文件实际存储名称、文件所属用户等）。HDFS 文件管理系统负责存储系统的上传文件及生成文件。

平台数据存储在服务器集群中，集群分为 NameNode 和 DataNode。NameNode 负责管理 HDFS 文件系统的原数据，处理客户端读写请求；DataNode 负责存储实际数据并执行数据块的读写。

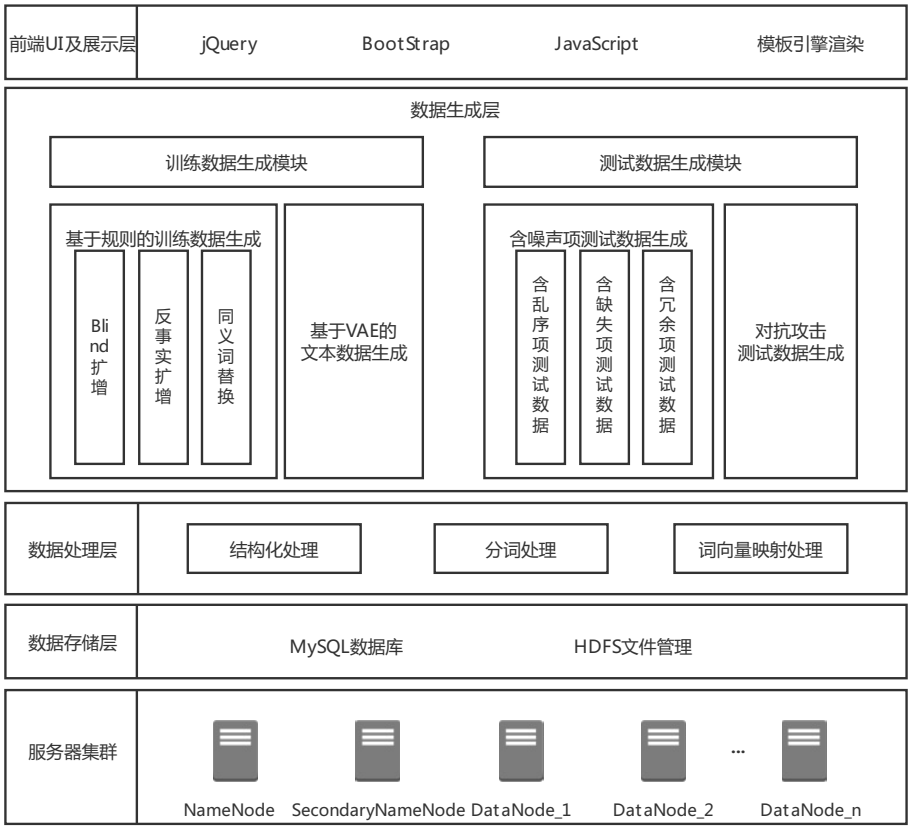


图 3.2: 司法文本数据自动化生成系统整体框架图

本文将利用“4+1”视图，通过多个视图维度进行模型的描述，捕捉系统要点。“4+1”视图包括场景视图、逻辑视图、进程视图、开发视图及物理视图。

场景视图为系统设计的起点，通过用户需求描述业务场景，使用 UML 用例图进行描述，本文的场景视图在上文 3.1 节的需求分析中已得到展现。

图 3.3是系统的逻辑视图，描述了系统所提供的功能与服务。其中，HTTPResponse 是系统进行 web 传输的基类，负责实现 web 通信中的基础功能。FileUpload 继承了 HTTPResponse 类，负责文件上传的功能。FileDownload 继承了 HTTPResponse 类，负责文件的下载功能。DataInteraction 类在系统中负责数据交互，与 HDFS 文件系统及 MySQL 数据库接口连接，负责文件及数据的存储与查找。HDFS 负责提供文件系统的读写文件服务。MySQL 负责提供数据库系统的读写与查询服务。DataPreprocess 类在系统中负责输入文件的预处理，包括文件的结构化、分词、词典映射等处理。TextGenerate 是系统进行数据生成的接口。分别被 TrainTextGenerate 类与 TestTextGenerate 类实现。其中，TrainTextGenerate 是系统进行训练数据生成的基类，负责训练数据的生成；其子类包括 RuleBasedTrainTextGenerate 及 VAETrainTextGenerate，分别负责基于规则的训练数据生成及基于 VAE 的训练数据生成。TestTextGenerate 是系统进行测试数据生成的基类，负责测试数据的生成；其子类包括 NoisyTestTextGenerate 及 AttackTestTextGenerate，分别负责含噪声测试数据的生成及对抗攻击测试数据的生成。

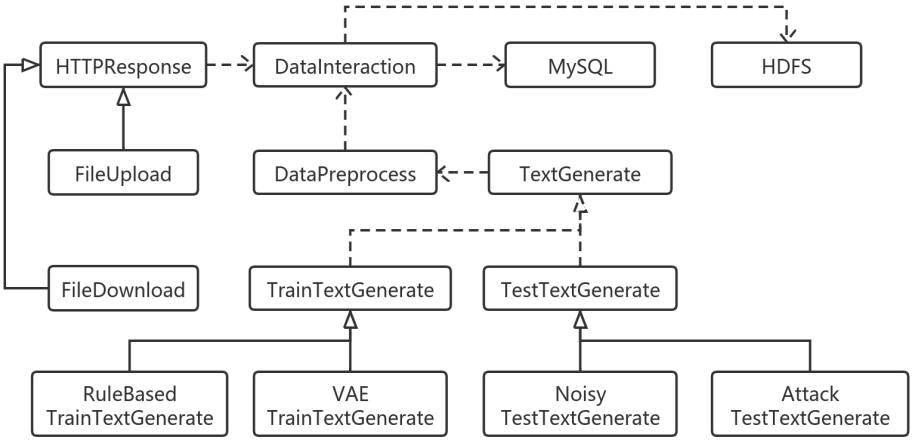


图 3.3: 司法文本数据自动化生成系统逻辑视图

图3.4是系统的开发视图，该视图从开发人员角度出发，描述了系统的静态组织结构。本系统由前端与服务端构成。前端负责系统与用户的交互，提供训练数据生成服务、测试数据生成服务及用户文件管理服务，传递用户提交的文件及数据至服务端，通过数据交互模块展示生成结果，并通过浏览器下载生成文件至本地。服务端负责系统逻辑的实现。服务端由数据交互模块、文本预处理模块、数据生成模块、用户文件管理模块四个模块组成。其中，数据交互模块在系统中负责数据及文件的交互，与 HDFS 文件系统及 MySQL 数据库连接，负责文件的存储、读取、下载及数据的存储与查找。文本预处理模块在系统中负

责文本文件的预处理操作，包括文件的结构化、分词、词典映射等预处理操作。数据生成模块是系统的核心模块，负责训练数据及测试数据的生成。

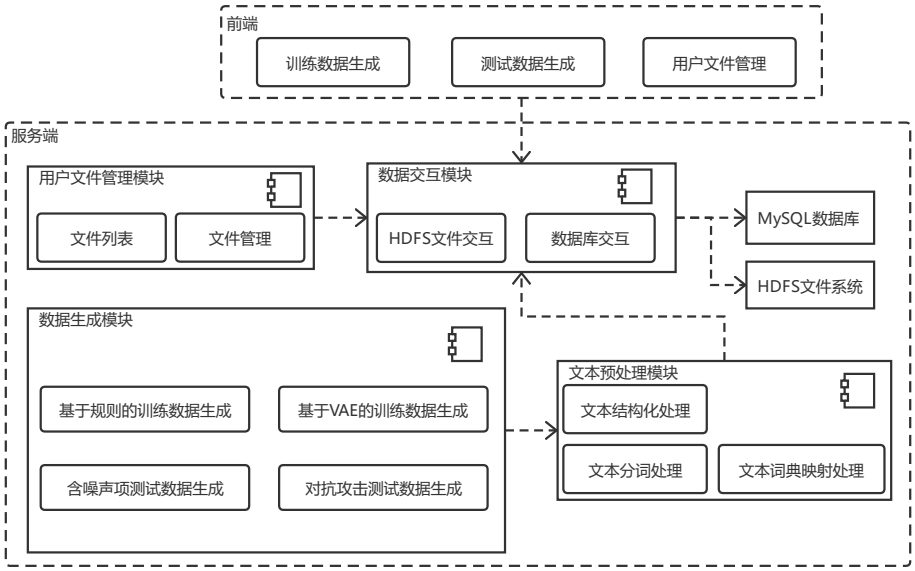


图 3.4: 司法文本数据自动化生成系统开发视图

图 3.5为系统的进程视图，进程视图从系统集成者角度出发，描述了系统的进程与通信。首先由用户提交待生成文件，系统调用 fileUpload 方法将用户的上传文件传入文本预处理模块，模块调用 DataPreprocess 方法进行文本的预处理操作，并将经过预处理的结构化输入文件通过 FileInteraction 方法传入 HDFS 文件系统中，将文件信息传入 MySQL 数据库中。当用户传入自定义的生成参数时，模块调用 getParam 方法将参数传入数据生成模块。系统将用户传入文件的存储位置、文件存储名称及文件在数据库中的键值传入数据生成模块，为模块提供输入文件的信息。数据生成模块调用 TextGenerate 方法，进行文件的生成，并将生成的文件通过 FileInteraction 方法传入 HDFS 文件系统中，将文件信息传入 MySQL 数据库中，将文件存储位置及文件存储名称传入 DataDownload 方法，将生成的文件通过浏览器下载至用户本地。当用户进行文件管理时，可通过 getFilelist 方法从数据库查看用户生成过的所有文件信息。

图 3.6是系统的物理视图，该视图从运维人员的角度出发，描述了系统的部署结构。用户通过用户机上的浏览器通过 HTTP 请求访问系统网站。Nginx 服务器通过代理转发功能解决了跨域问题，使用户访问到应用服务器，系统应用服务器由 Django 框架实现，Django 框架下的 WSGI 服务器处理用户 HTTP 请求解

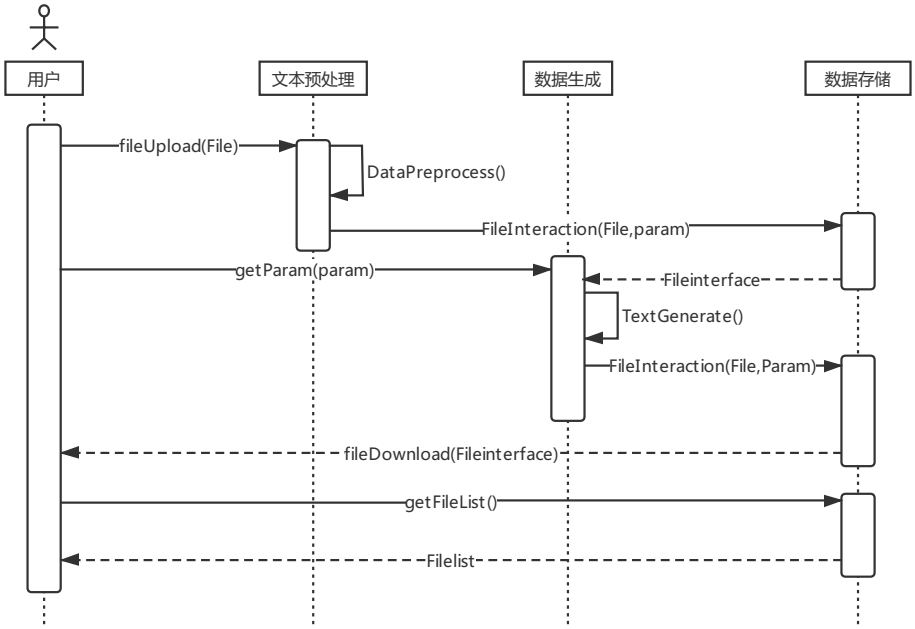


图 3.5: 司法文本数据自动化生成系统进程视图

析，将 URL 地址映射到匹配的视图函数，通过视图函数处理内部业务逻辑，返回 HTTP 响应。系统在数据服务器部署 MySQL 应用，用于承担系统的数据存储及查询服务，系统与数据服务器通过 TCP 协议进行传输；并在集群中部署 HDFS 应用，用于承担系统的文件存储服务，系统与 HDFS 集群通过 SSH 协议进行传输。

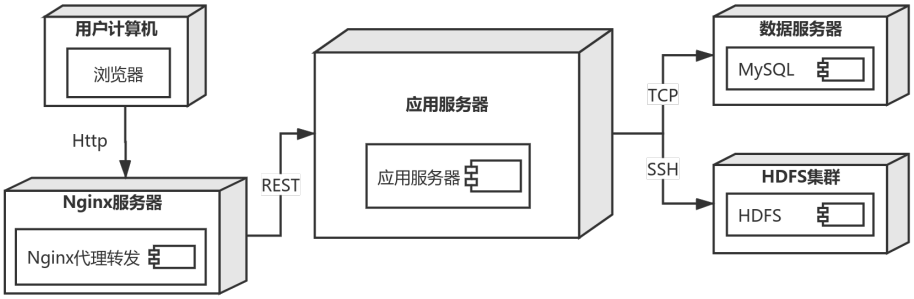


图 3.6: 司法文本数据自动化生成系统物理视图

依据上述分析，平台基本功能模块图如图 3.7所示，包括数据交互模块、用户文件管理模块、文本预处理模块、数据生成模块。其中，用户生成模块为系统核心，共包含四种类型的生成方式，包括基于规则的训练数据生成、基于 VAE

的训练数据生成、含噪声项测试数据生成、对抗攻击测试数据生成。

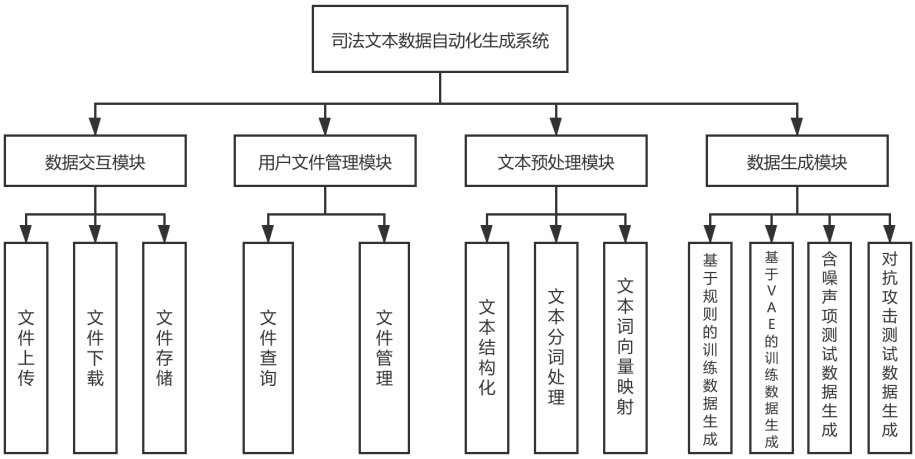


图 3.7: 司法文本数据自动化生成系统功能模块图

3.3　数据交互模块设计

数据交互模块负责用户与系统间的数据及文本的交互，将用户、应用系统、MySQL 数据库系统、HDFS 文件管理系统进行了连接。模块功能包括存储上传的用户文件、查询用户数据、将生成数据传入页面展示以及下载文件至用户浏览器等。数据交互模块的类图如图 3.8所示，具体实现的类和接口包括 HTTPResponse、FileUpload、FileDownload、DataService、FileInteraction、FileInteractionHdfs、FileInteractionMySQL。其中，HTTPResponse 是系统进行 web 传输的基类，实现 web 通信中的基础功能。FileUpload 继承自 HTTPResponse 类，实现文件的上传功能，FileDownload 同样继承自 HTTPResponse，负责文件的下载功能。DataService 类是 web 传输的数据及文件与 MySQL 数据库系统、HDFS 文件系统的传输介质，负责将用户传输的数据及文件传递给 MySQL 与 HDFS，也负责将用户传入前端的参数传递到业务逻辑代码端。FileInteraction 为系统与数据库系统交互的接口，分别被 FileInteractionHdfs、FileInteractionMySQL 类实现。其中，FileInteractionHdfs 负责应用系统与 HDFS 系统的交互，提供了与 HDFS 交互的接口函数，包括文件读取、文件写入、文件存储、文件下载等；FileInteractionMySQL 负责应用系统与 MySQL 数据库系统的交互，提供了系统进行数据写入、数据查询的函数。

在文件存储至 HDFS 时会存在文件的重复问题，系统在设计时应尽量避免相同文件的重复存储，以免造成存储资源的浪费。系统在设计时引入信息摘要

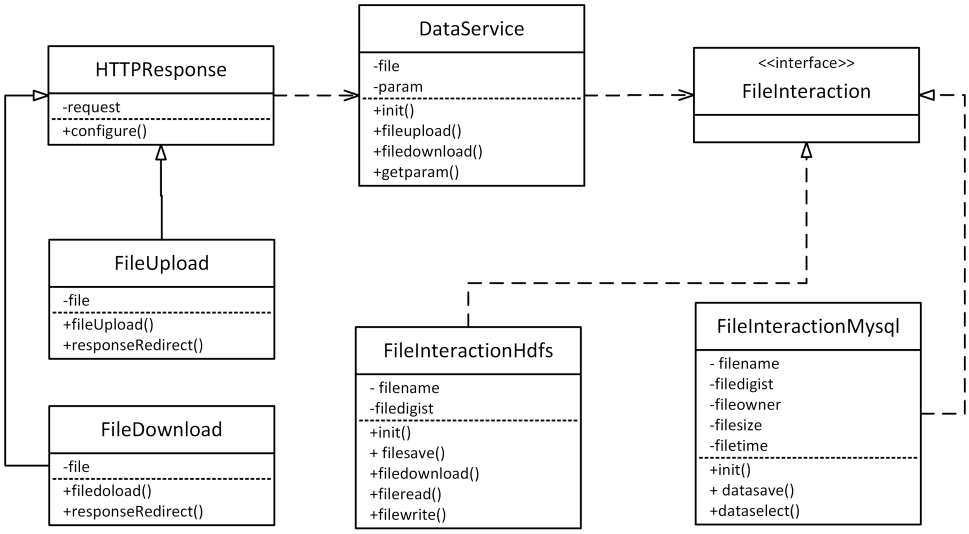


图 3.8: 数据交互模块类图

的概念。信息摘要指利用信息摘要算法，对信息内容进行数据变换，得到特征码。特征码不可逆，具有高度的离散性，而且由于特征码长度较长，任意信息之间具有相同特征码的可能性极低。

因此，系统将以摘要特征码作为文件在存储时的文件名，当有相同文件内容的文件输入时，将与文件系统内已有的文件名重复，系统将不再重复存储。系统设定了数据表 Link，用来记录以该文件名为信息摘要的文件数量，相同内容的文件将共享同一个存储文件，而文件信息则存储在文件信息表 File 中。当用户删除文件时，若 Link 中的链接数量为 1 时，说明以该文件名为信息摘要的文件数量为 1，系统可以直接删除该文件；若 Link 中的链接数量大于 1 时，说明以该文件名为信息摘要的文件数量大于 1，不可直接删除文件，只能将链接数量减 1。

如图 3.9为存储文件流程图。系统在文件输入后，计算其信息摘要，得到特征码 digist 值为 A。digist 将作为文件在 HDFS 文件系统中的存储名称。系统具有数据表 File，用来存储文件相关信息，包括文件的存储名。查询 File 表中是否已有 digist 值为 A，如没有，则将文件以 A 为存储名，存储在 HDFS 文件系统中，并将文件信息存储在 File 表内。系统链接表 Link 添加 digist 值为 A 的数据行，其链接数为 1，表示存储文件名为 A 的文件数量为 1。若 File 表中已有 digist 值为 A，表示输入文件内容为重复文件，因此系统将不再上传该文件，直接将文件信息存储在 File 表中，并将 Link 表中 digist 值为 A 的链接数加 1, 表示以 A 为存储名的文件数量增加 1。

如图 3.10为删除文件流程图。用户删除文件时，查询文件表 File，获取该文

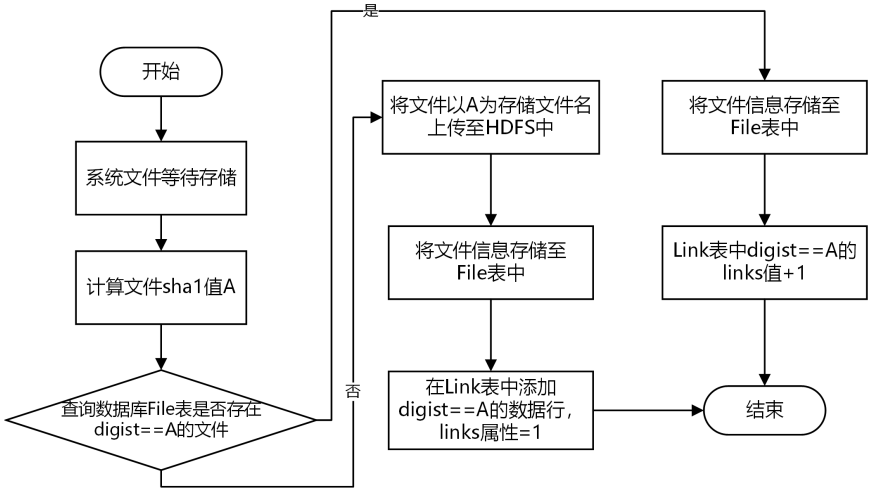


图 3.9: 系统存储文件流程图

件的存储名称 A。然后查询 Link 表，获取 digist 为 A 的链接数量。若链接数量为 1 时，说明以该文件名为信息摘要的文件数量为 1，系统可以直接删除该文件；若 Link 中的链接数量大于 1 时，说明以该文件名为信息摘要的文件数量大于 1，不可直接删除文件，只能将链接数量减 1。

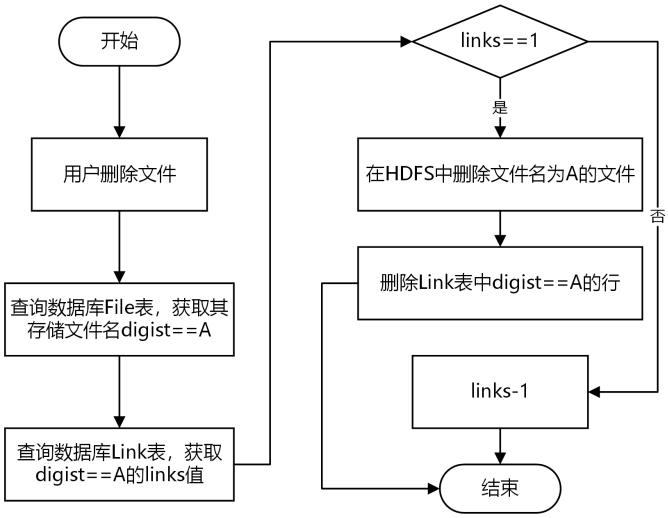


图 3.10: 系统删除文件流程图

3.4　文本预处理模块设计

文本预处理模块负责对输入的文本文件进行生成前的预处理操作，避免生成时相同文件的反复预处理，造成时间和计算资源的浪费。文本预处理模块类图

如图 3.11所示。具体实现的类包括 DataPreprocess、TextStructured、WordIndexMapping、TextSegmentation。其中，DataPreprocess 为模块的基类，负责实现预处理需要的文件交互，定义了输入输出文件名等变量，以及读取文件、写入文件、存储至 HDFS 文件系统、写入文件信息至数据库等函数。TextStructured、WordIndexMapping、TextSegmentation 类是 DataPreprocess 的子类，分别负责三项独立的预处理任务。TextStructured 类定义了 structured 函数，负责文本的结构化；TextSegmentation 类定义了 segment 函数，负责文本的分词处理；WordIndexMapping 类负责词典序号映射，create_dictionary_matrix 函数构建语料库词典，texts_to_index 函数将文本映射为语料库词典序号。

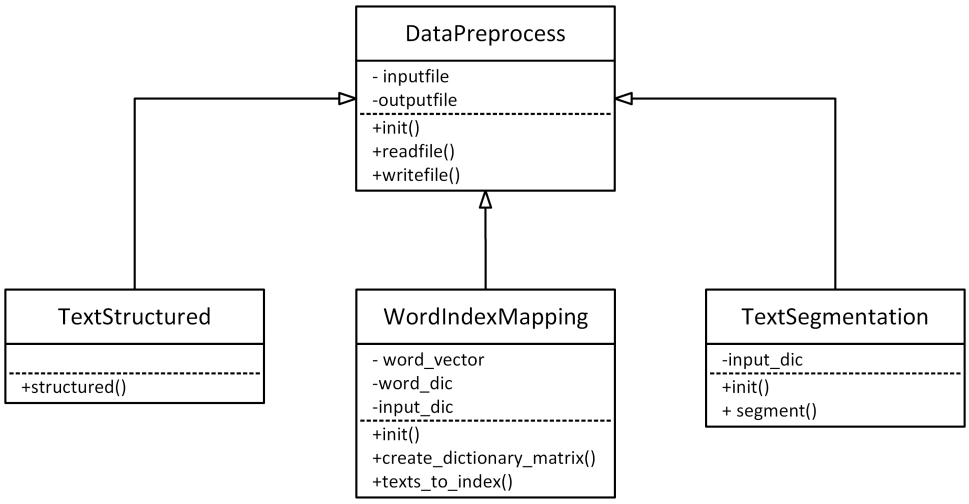


图 3.11: 文本预处理模块类图

文本预处理的流程图如图 3.12所示。用户传入待生成的文本文件，模块检查输入的文件格式是否符合系统的要求，若不符合则需重新输入。若符合系统要求，将开始进行预处理工作。首先，系统创建 JSON 文件，用来存储经过结构化的文本。系统将解析用户输入的文本文件，将文本内容、标签、文本序号结构化并分开存储，写入 JSON 文件。接下来进行文件的分词操作，并将分词结果存储在 JSON 文件中。接下来，将写好的 JSON 文件的文件信息写入 MySQL 数据库中，包括文件名、文件归属用户、文件在 HDFS 系统中的存储名、文件大小、文件存储时间等。下一步，将文件按照定义的文件存储位置及存储名称，存储在 HDFS 文件系统中。当用户首次选择需要使用词典的生成方式时（包括基于 VAE 的训练数据生成方式、对抗攻击测试数据生成方式），系统将词典映射写入 JSON 文件，并将文件信息写入 MySQL 数据库，数据存储在 HDFS 中。当用户进行数据生成操作时，系统将从 HDFS 系统加载结构化的 JSON 文件，从文件中提取所需的输入文本信息，避免了预处理过程的重复计算。

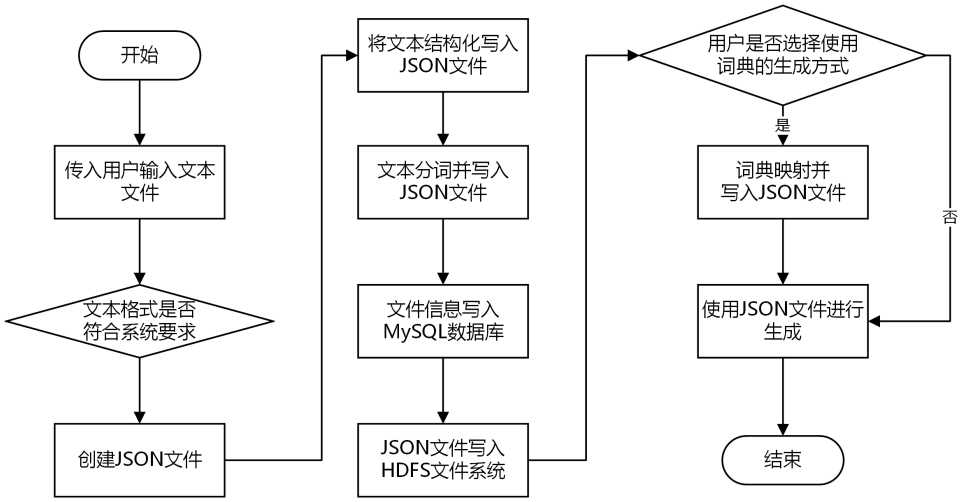


图 3.12: 文本预处理模块流程图

3.5　数据生成模块设计

数据生成模块是系统的核心功能，它可以为用户提供全面的司法文本数据生成服务，分为训练数据生成及测试数据生成两部分。训练数据生成包括基于规则的数据生成方式与基于 VAE 的数据生成方式。其中，基于规则的数据生成方式有 blind 扩增、反事实扩增、同义词替换。测试数据生成包括含噪声测试数据的生成与对抗攻击测试数据的生成。其中，含噪声测试数据的生成方式有含乱序项生成方法、含缺失项生成方法及含冗余项生成方法。数据生成模块功能图如图 3.13所示。

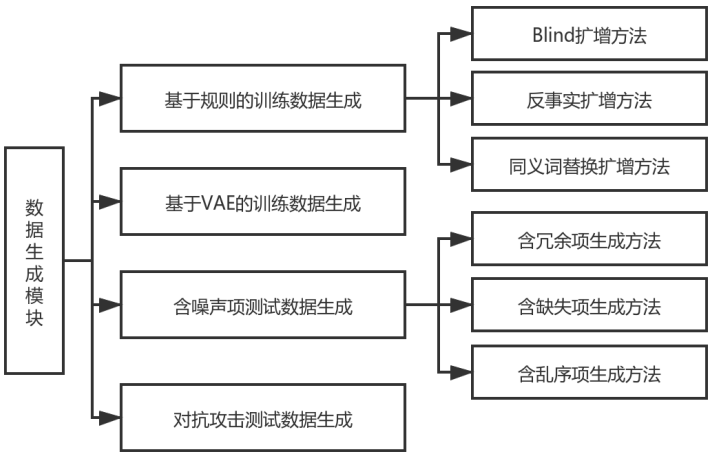


图 3.13: 数据生成模块功能图

数据生成模块类图如图 3.14所示。其中，TextGenerate 为模块接口，定义了数据生成过程中的通用功能接口，包括与 HDFS 文件系统的交互、读写文件等接口函数。TrainTextGenerate 类与 TestTextGenerate 类实现了 TextGenerate 接口，分别为训练数据生成的基类与测试数据生成的基类，实现了数据生成所需的基础功能函数。

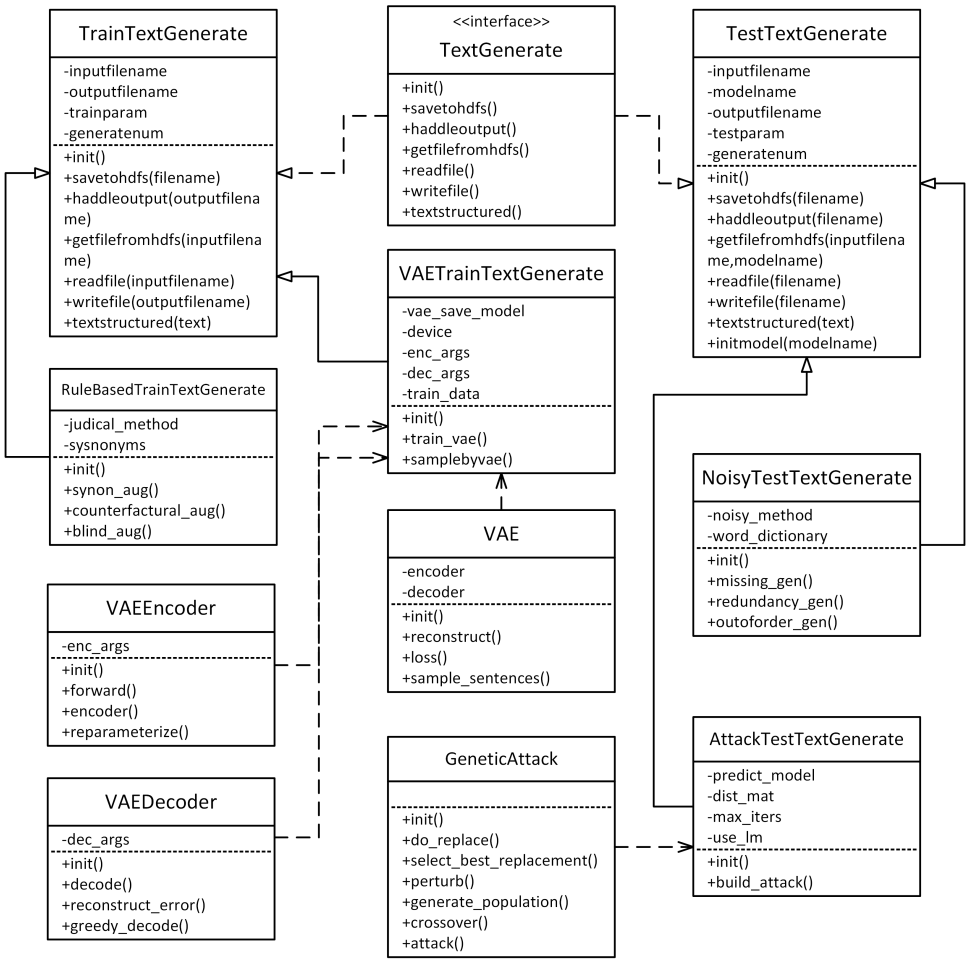


图 3.14: 数据生成模块类图

RuleBasedTrainTextGenerate、VAETrainTextGenerate 类继承了 TrainTextGenerate 父类。RuleBasedTrainTextGenerate 负责基于规则的训练数据生成的计算，其中，synon_aug()、counterfactural_aug()、blind_aug() 方法分别定义了同义词替换扩增方法、反事实扩增方法、blind 扩增方法。基于规则训练数据生成的活动图如图 3.15所示。在进入生成页面后，用户首先需要传入待生成的训练数据文件，系统 session 会保留最后一次上传的文件信息。在文件上传后，系统在数据交互模块会判定文件是否符合系统规范，若符合将开始文本的预处理。当系统传入

用户自定义生成参数，且文本预处理结束后，系统开始生成操作。系统对用户选择的生成方法进行多线程并发执行，生成 json 格式的结构化文件，包括生成结果、利用的生成方法、文本序号等字段。当用户选择的多种生成方法全部执行结束后，模块将对生成结果排序后存储在文件系统中，生成结果将展示在页面，并提供下载接口。

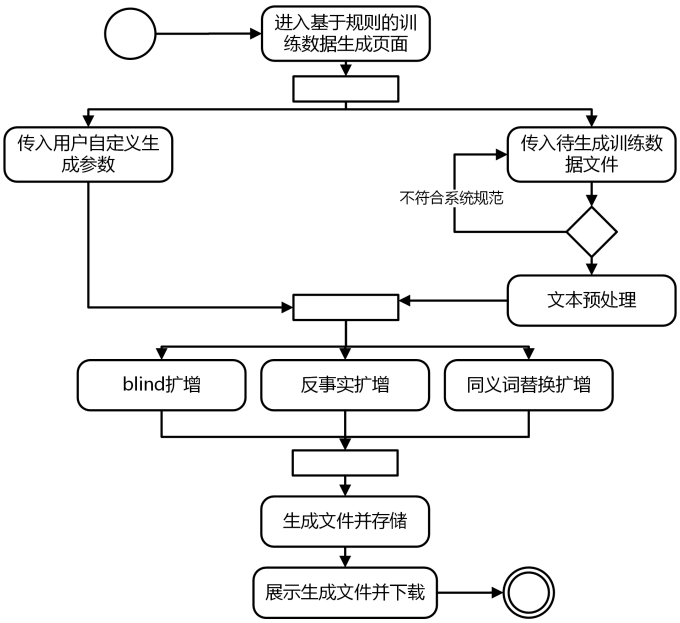


图 3.15: 基于规则的训练数据生成活动图

VAETrainTextGenerate 负责基于 VAE 的训练数据生成，定义了关于 VAE 模型结构、超参数等变量，train_vae 函数负责训练 VAE 模型、samplebyvae 函数负责利用经过训练的 VAE 模型进行数据生成。在 VAETrainTextGenerate 类中实例化了 VAE、VAEDecoder、VAEEncoder 三个对象，以上三个类中定义了 VAE 模型的模型架构及模型优化函数等与 VAE 模型相关的方法及变量。基于 VAE 的训练数据生成活动图如图 3.16所示。在进入生成页面后，若用户没有已训练完成的 VAE 模型，需要首先进行 VAE 模型的训练。用户需要提交训练所需文本文件，以及自定义的训练超参数。系统将根据用户的提交数据及参数，通过系统调用的方式开始模型训练，并在数据库记录模型文件信息及状态。当模型训练完成后，将在数据库更新模型状态为可用。当用户已有训练完成的 VAE 模型时，可以查看模型详情，并选择模型进行数据生成。提交生成数量后，系统将在文件系统中寻找模型文件，加载模型文件进行数据生成。生成结果将存储并展示在前端界面，并提供下载接口。

NoisyTestTextGenerate、AttackTestTextGenerate 类继承了 TestTextGenerate 父

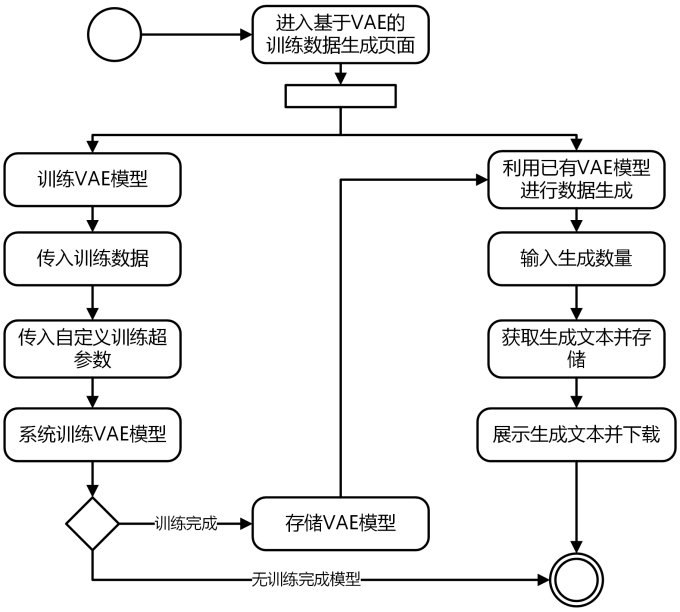


图 3.16: 基于 VAE 的训练数据生成活动图

类。NoisyTestTextGenerate 负责含噪声项测试数据生成，其中，missing_gen()、redundancy_gen()、outoforder_gen() 方法分别定义了含缺失项生成方法、含冗余项生成方法、含乱序项生成方法。含噪声项数据生成的流程与基于规则的训练数据生成相似，在传入待生成的测试文件后开始文本预处理流程。当传入生成参数且预处理结束后，对用户选择的生成方式进行多线程并发执行。生成结果将存储在文件系统，并提供下载接口。

AttackTestTextGenerate 类负责对抗攻击测试数据的生成，加载输入的待攻击模型与测试数据，将测试数据与用户自定义参数逐条实例化为 GeneticAttack 对象，进行对抗攻击测试数据生成。GeneticAttack 类中定义了对抗攻击的各项函数。由于对抗攻击测试数据生成的过程需要接受对抗攻击预测模型的参与，因此系统需要输入的文件包括：测试数据、预测模型以及训练集词表。系统目前支持基于 fasttext 训练的以.bin 为后缀名的预测模型文件，以及基于 tensorflow 训练的以.pb 为后缀名的预测模型文件，支持多分类多标签预测模型。训练集词表用于将测试集中的词语转化为预测模型输入。输入文件均需要按照系统规定的格式，系统将检查输入文件，如果不符合要求将重新上传，对抗攻击测试数据生成活动图如图 3.17所示。

3.6　用户文件管理模块设计

用户文件管理模块负责对用户在系统中生成过的文件进行管理，模块活动图如图 3.18所示。用户登录个人中心进行文件管理，首先在数据库中进行检查，

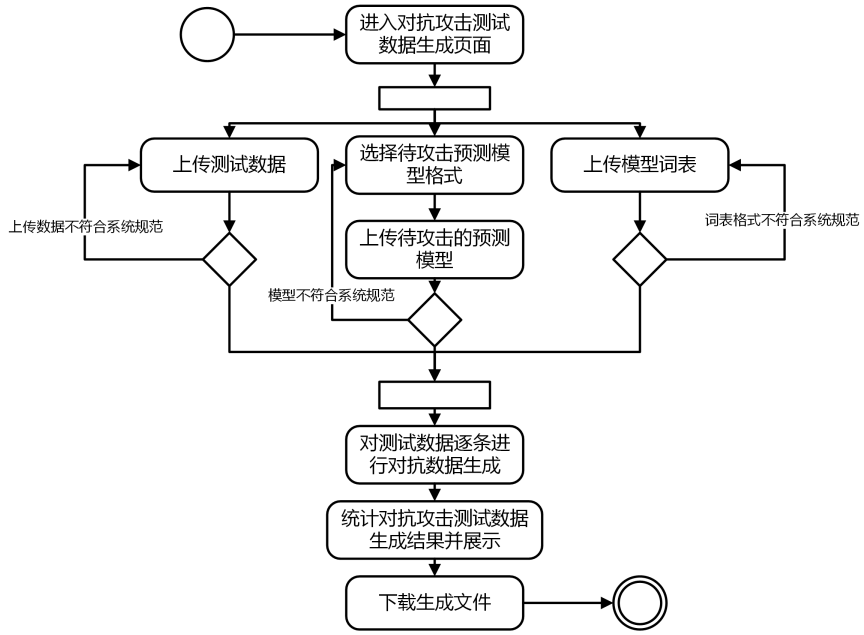


图 3.17: 对抗攻击测试数据生成活动图

用户是否在系统中有未删除的生成文件，若无文件则显示空白列表，若有文件，则显示用户生成的文件列表，列表展示了文件名、文件大小、文件生成日期等基本信息，并提供了删除文件与下载文件的选项。若用户选择下载文件，系统在数据库中查询文件在 HDFS 文件系统中的存储位置，并通过浏览器下载文件至用户本地。若用户选择删除文件，系统将首先在数据库中查询文件在 HDFS 系统中的存储位置，接着删除 MySQL 数据库中的文件信息，并删除 HDFS 文件系统中的存储文件，避免造成文件系统中无用文件的大量存储。

3.7　数据库设计

本节介绍系统中所涉及到的数据库结构。在系统的文件管理过程中，需要对文件信息进行存储，以便在进行文件交互与文件管理时，可以迅速地从 HDFS 文件系统中寻找到文件名称及所在位置。本节根据概念模型，生成实体关系 ER 图，表示实体中的各个属性信息，以及实体间的关联关系。并利用数据库表进行实体属性的详细说明。

如图 3.19中的 ER 图表示了系统文件管理数据库的存储关系。系统文件管理的数据库设计围绕文件信息表 File 展开。文件信息表 File 存储了文件的重要信息，包括文件 id file_id、文件名 file_name、文件存储名 digist、文件所属用户 owner、文件大小 size、文件类型 kind（表示文件是由哪种生成方式生成的）、文件生成时间。用户表 User 存储系统用户的基本信息，与 File 表是 1：N 的包含

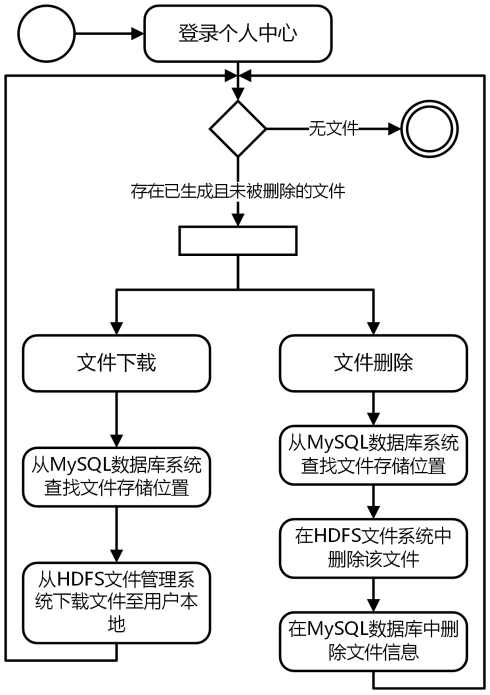


图 3.18: 用户文件管理活动图

关系，一个用户可拥有多个文件。User 表包括用户 id user_id、用户名 user_name 等用户注册信息。链接表 Link 表示以 digist 为存储名的文件个数，包括文件存储名 digist、文件链接数 links，与 File 表是 1：M 的包含关系。

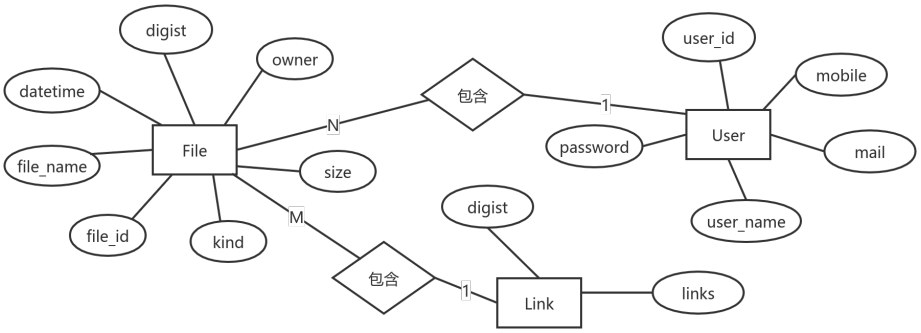


图 3.19: 系统文件管理数据库实体关系图

如图 3.20中的 ER 图表示了系统 VAE 模型管理数据库的存储关系。系统 VAE 模型管理的数据库设计围绕模型集合表 Models 展开。模型集合指用户提供的训练数据中包含多个标签类型时，所有标签类型对应训练的 VAE 模型的集合。模型集合表 Models 存储了模型集合的重要信息，包括模型集合 id model_id、模型集合名称 model_name、模型集合所属用户 owner、模型集合状态 status、模型

集合创建时间、模型集合相关参数信息等。Case 表存储了模型集合下对应的各标签的具体模型信息，包括模型 id case_id、模型名称 case_name、模型所属集合 id model_id、模型存储位置 case_dir。Models 表与 Case 表是 1：N 的包含关系，一个模型集合中可以有多个标签模型。Figure 表存储了模型训练过程中生成的损失函数图及困惑度函数图的图片信息，包括图片 id、图片名称、图片类型、所属模型集合 id 等信息。Models 表与 Figure 表是 1：M 的包含关系，一个模型集合中可以有多个图片。

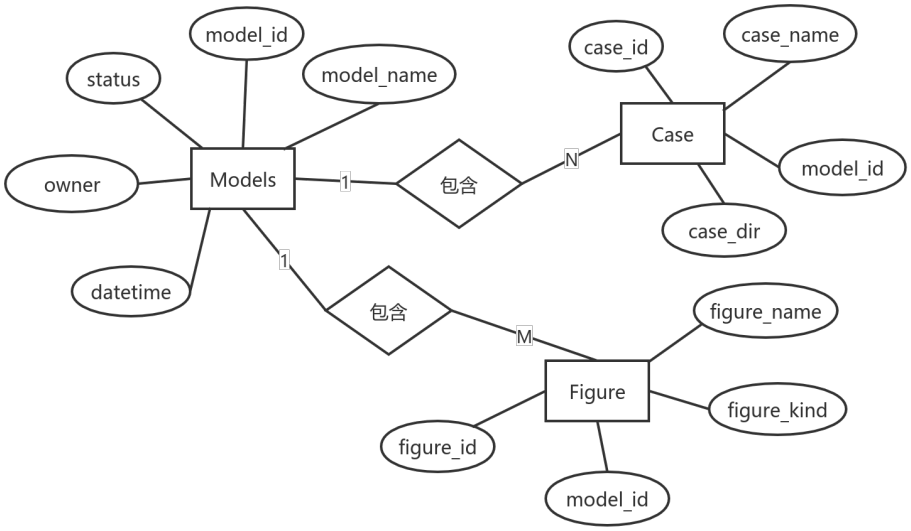


图 3.20: 系统 VAE 模型管理数据库实体关系图

如图 3.21中的 ER 图表示了系统中地理位置数据库的存储关系。地理位置数据库在反事实数据生成方式中被应用，目的为快速的将省份、城市、以及区县关系进行对应查找。省份信息表 Province 存储了省份名称及 ID，城市信息表 City 存储了城市名称、城市 ID 及城市所属的省份 ID，与 Province 表为 1：N 的包含关系。区县信息表存储了区县名称、区县 ID 及区县所属的城市 ID，与 City 表为 1：M 的包含关系。

数据库表结构设计如下所述：

文件信息表 File：如表 3.8所示，File 表中存储了与用户文件相关的信息，包括文件 id file_id、文件名 file_name、文件存储名 digist、文件所属用户 owner、文件大小 size、文件类型 kind（表示文件是由哪种生成方式生成的）、文件状态 status、文件生成时间 datetime。

用户信息表 User：如表 3.9所示，User 表存储了用户的注册信息，包括用户 id user_id、用户名 user_name、用户密码 password、用户邮箱、用户手机号。

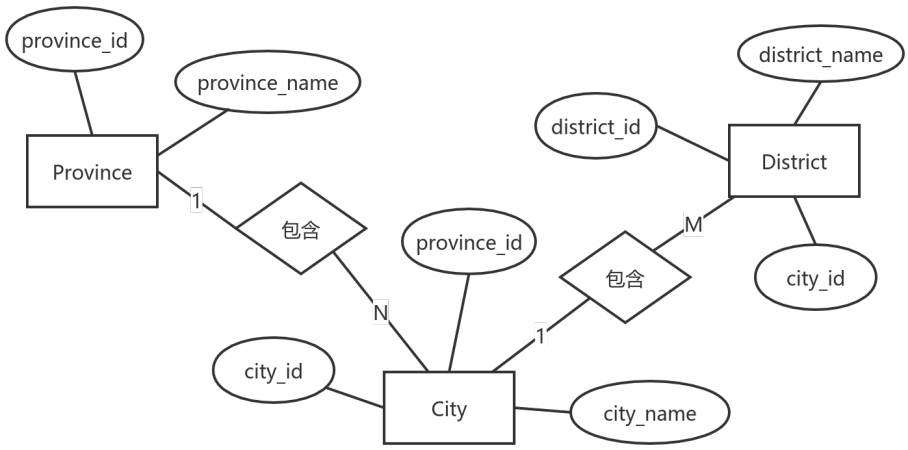


图 3.21: 系统地理位置数据库实体关系图

表 3.8: 文件信息表 File 表结构

字段名	类型	允许为空	说明
file_id	bigint(20)	否	文件号
file_name	varchar(64)	否	用户文件名
digist	varchar(64)	否	文件存储名
owner	varchar(64)	否	文件所属用户
size	bigint(20)	否	文件大小
kind	varchar(64)	否	文件类型
datetime	varchar(64)	否	文件存储时间

表 3.9: 用户信息表 User 表结构

字段名	类型	允许为空	说明
user_id	bigint(20)	否	用户号
user_name	varchar(64)	否	用户名
password	bigint(20)	否	用户密码
mail	varchar(64)	否	用户邮箱
mobile	bigint(20)	是	用户手机号

文件链接表 Link：如表 3.10所示，文件链接表 Link 存储了以 digist 为文件存储名的文件个数，包括文件存储名 digist、文件链接数 links。

表 3.10: 文件链接表 Link 表结构

字段名	类型	允许为空	说明
digist	varchar(64)	否	文件存储名
links	bigint(20)	否	文件链接数

模型集合信息表 Models：如表 3.11所示，Models 表中存储了与 VAE 模型集合相关的信息，将每次训练数据下各标签生成的所有模型进行统一管理。其中，模型集合状态 status 标记系统生成 VAE 模型的进度。当提交模型生成任务时，标记为“waiting”；当模型生成中出错，修改字段为“error”；当模型生成完成，修改字段为“success”。

表 3.11: 文件集合表 Models 表结构

字段名	类型	允许为空	说明
model_id	bigint(20)	否	模型集合号
model_name	varchar(64)	否	模型集合名称
owner	varchar(64)	否	模型集合所属用户
status	varchar(64)	否	模型状态
datetime	varchar(64)	否	模型集合创建时间
ppl_value	varchar(64)	否	模型集合平均迷惑度
traindata_name	varchar(64)	否	模型集合的训练数据名称
testdata_name	varchar(64)	否	模型集合的测试数据名称
validdata_name	varchar(64)	否	模型集合的验证数据名称
param_epoch	varchar(64)	否	超参数：训练时的最大 epoch 数
param_klstart	varchar(64)	否	超参数：KL 散度权重起始值
param_lrspeed	varchar(64)	否	超参数：KL 散度权重恢复到 1 的 epoch 数
param_klchange	varchar(64)	否	超参数：更换学习率的 epoch 个数
param_lrchange	varchar(64)	否	超参数：学习率下降速度

具体模型信息表 Case：如表 3.12所示，Case 表中存储了模型集合中各标签下的具体模型信息。包括模型名称（即标签名称）、模型在文件系统中的存储文件名等信息。

表 3.12: 具体模型信息表 Case 表结构

字段名	类型	允许为空	说明
case_id	bigint(20)	否	模型号
case_name	varchar(64)	否	模型名称
case_dir	varchar(64)	否	模型存储名称
model_id	bigint(20)	否	模型所属模型集合号
train_num	bigint(20)	否	该标签下训练数据数量
test_num	bigint(20)	否	该标签下测试数据数量
valid_num	bigint(20)	否	该标签下验证数据数量

图片信息表 Figure：如表 3.13所示，Figure 表中存储了模型训练过程中生成的损失函数图及困惑度函数图的图片信息，包括图片 id、图片名称、图片类型、所属模型集合 id 等信息。

表 3.13: 图片信息表 Figure 表结构

字段名	类型	允许为空	说明
figure_id	bigint(20)	否	图片号
figure_name	varchar(64)	否	图片名称
figure_kind	varchar(64)	否	图片类型
model_id	bigint(20)	否	图片所属模型集合号

省份信息表 Province：如表 3.14所示，省份信息表 Province 存储了国内的省份名称，包括省份号 province_id、省份名称 province_name。

表 3.14: 省份信息表 Province 表结构

字段名	类型	允许为空	说明
province_id	bigint(20)	否	省份号
province_name	varchar(64)	否	省份名称

城市信息表 City：如表 3.15所示，城市信息表 City 存储了国内的城市名称及其包含关系，包括城市号 city_id、城市名称 city_name、该城市所属省份 province_id。

表 3.15: 城市信息表 City 表结构

字段名	类型	允许为空	说明
city_id	bigint(20)	否	城市号
city_name	varchar(64)	否	城市名称
province_id	bigint(20)	否	省份号

区县信息表 District：如表 3.16所示，区县信息表 District 存储了国内的区县名称及其包含关系，包括区县号 district_id、区县名称 district_name、该区县所属城市 city_id。

表 3.16: 区县信息表 District 表结构

字段名	类型	允许为空	说明
district_id	bigint(20)	否	区县号
district_name	varchar(64)	否	区县名称
city_id	bigint(20)	否	城市号

3.8　本章小结

本章首先将系统分为用户文件管理模块、数据交互模块、文本预处理模块以及数据生成模块，并对以上四个模块做了详细的需求分析，描述了系统的基

本功能。然后对系统进行了总体设计，以不同视图对系统进行了多角度的描述。并对系统的各个模块进行了详细的设计，描述了模块的运行流程及开发类图。最后叙述了系统的数据库设计。

第四章　系统实现

本章介绍了司法文本数据自动化生成系统在文本预处理、数据生成、数据交互、用户文件管理模块的核心实现，以及 Hadoop 高可用分布式配置。介绍了各模块在流程及算法方面的实现，分析并展示了司法文本数据的生成结果，最后对平台的核心功能进行了测试。

4.1　文本预处理

文本预处理是系统进行数据生成的前提步骤，包括文本结构化处理、文本分词处理及语料库词典映射。其目的是在文本文件输入时进行结构化处理，使输入文件以结构化形式存储。并将文本预处理结果进行存储，避免相同文件的多次计算，造成时间和资源的浪费。经过预处理的结构化文本以 json 格式存储在文件中，并将文件存储在 HDFS 文件系统中。结构化文本内容包括文本序号、原始文本、文本标签、分词后文本、文本词典映射列表，各部分功能及格式如表 4.1所示。图 4.1是本模块中文本结构化处理的核心代码。

表 4.1: 文本预处理结构化文本格式

字段名称	格式	说明
id	int	存储此条文本的序号
sentence	string	存储用户输入的此条司法文本的原始格式
label	string	存储用户输入的此条司法文本具有的标签
seg_sentence_list	list	将用户输入的此条司法文本经过分词处理后，以 list 格式存储在结构化文本中
word_index_list	list	将分词后的此条司法文本与语料库字典对应，将词语映射为字典 index 存储在结构化文本中

系统在分词处理过程中调用了 jieba 分词器在 python 上的 API 接口。在语料库词典映射过程中，基于 VAE 的生成方法需要首先建立语料库。基于分词结果对训练数据集中的词语做词频统计，按词频大小优先选择 n 个词语作为语料库。对抗攻击生成方法由用户提供语料库文件。词典映射时，将文本中的词语依次在语料库中查找，返回词典序号，若词典中不存在该词语，返回“unknow”标记。核心代码如图 4.2所示。

```
class TextStructured(DataPreprocess):
  def __init__(self, input_file):
    self.inputfile = input_file
    self.line_dict = {}
  ...
  def structured(self):
    sentence_list = super().readfile(self.inputfile)
    for index, line in enumerate(sentence_list):
      self.text_structured_byline(line, index)
    super().writefile(self.line_dict)
  #文本结构化，添加文本序号，sentence标签与label标签
  def text_structured_byline(self, line, index):
    line_list = line.strip().split('\t')
    self.line_dict['index'] = index
    self.line_dict['sentence'] = line_list[0]
    self.line_dict['label'] = line_list[1]
  ...
```

图 4.1: 文本结构化处理核心代码

```
class WordVectorMapping(DataPreprocess):
  ...
  #建立语料库
  def create_dictionary_matrix(self, words_list):
    self.tokenizer = Tokenizer()
    self.tokenizer.fit_on_texts(words_list)
  #语料库词典映射，写入word_index_list标签
  def texts_to_index(self,input_text):
    input_seqs = self.tokenizer.texts_to_sequences(input_text)
    self.line_dict['word_index_list'] = input_seqs
  ...
```

图 4.2: 语料库词典映射核心代码

4.2　数据生成

数据生成是本系统的核心功能，负责利用用户的输入文本，生成具有不同功能的文本。根据用途的不同，系统将数据生成模块分为两大类：训练数据生成与测试数据生成。本节介绍了各个数据生成方式的具体实现。

4.2.1　基于规则的训练数据生成

基于规则训练数据生成的目的是为司法领域人工智能系统提供更广阔的训练数据，因此生成的数据需要与原数据具有相同的数据分布，不改变其语法及语义。系统中基于规则的生成方式共三种：Blind 扩增方法、反事实扩增方法、同

义词替换扩增方法。

Blind 扩增方法的具体实现为：在司法文本中存在一部分与案情审判具有弱相关性的特征性词汇，包括：年份、地点、涉案人员等。这类词汇在司法文书中具有较高的出现频率，且具有丰富的表达方式。但在人工智能系统对司法文书进行分析时，例如刑期预测、罪名预测等情形下，分析结果与此类词汇不具有相关性。因此，Blind 扩增方法的基本思想为在不改变语法语义的情况下，使用标准化的标签，来替代某种类型词汇。例如：省份、城市、区县、年份等，具体标签如表 4.2所示。

表 4.2: Blind 扩增方法标签说明

词汇类型	标签名称	示例说明
省份	Province	将陕西省、湖南省等省份名称替换为 Province 标签
城市	City	将北京市、咸阳市等城市名称替换为 City 标签
区县	County	将丰台区、礼泉县等区县名称替换为 County 标签
年份、月份、日期、时间	Year、Month、Day、Time	例如：将 2019 年 3 月 13 日 10 时许替换为：Year 年 Month 月 Day 日 Time 时许
涉案人员	Name	将张某、王某等涉案人员名称替换为 Name 标签

反事实扩增方法的具体实现为：反事实扩增方法改变的对象与 Blind 扩增方法相同，同样是与案情审判具有弱相关性的特征词汇，包括：年份、地点、涉案人员等。但与 Blind 扩增方法有所不同，反事实扩增方法使用与事实相反的词汇代替原有词汇，改变词语含义，却不改变整体语义的表达。例如：替换地点名称、替换涉案人员名称等操作。Blind 扩增由于每种类型词汇只拥有一个标签，因此生成的文本数量与输入文本数量保持一致，即单倍扩增。与 Blind 扩增相比较，反事实扩增由于每种类型词汇具有多种反事实表达方式，可以产生数量更多的生成文本，系统在生成前提供参数，用于用户选择生成文本数量对于输入文本数量的倍数。

在替换地点时，系统内置了全国的省份名、城市名及区县名，以及各级地名的从属关系，存储在数据库中。在替换时，随机选择最高级别地名，并从数据库中查找其下属地名并随机选择其中之一，如图 4.3为地名替换的核心计算方法。

同义替换扩增方法具体实现为：顾名思义，同义词替换方法将原始文本中的同义词进行合理替换，生成新文本。由于法律术语具有专业性与准确性，因此在替换时，系统选择了经过筛选的中性同义词键值对，存储在数据库中。在生成数据时系统根据用户给出的修改比例参数，从数据库中查询同义词键值对，替换原始文本生成新文本。

```
def counterfactural_aug(self):
    words = pseg.cut(fact)
    new_fact = ""
    cflag = 0
    for word, flag in words:
        ...
        if flag in ['ns', 'nr']:
            if "县" in word or "区" in word:
                sql = 'select area.name from area join city ON area.cityCode = city.code ' \
                      'join province ON area.provinceCode = province.code \
                      where city.name="%s";' % city_name
                self.cursor.execute(sql)
                word = random.choice(self.cursor.fetchall())
                area_name = word
            elif "市" in word:#若该地名为直辖市，从市级单位开始选择
                if province_name:
                    sql = 'select city.name from city ' \
                          'join province ON city.provinceCode = province.code \
                           where province.name="%s";' % province_name
                    self.cursor.execute(sql)
                    word = random.choice(self.cursor.fetchall())
                else:
                    word = self.city[random.randint(0, self.province.__len__() - 1)]
                city_name = word
            elif "省" in word:#从高级别地名单位开始随机选择
                word = self.province[random.randint(0, self.province.__len__() - 1)]
                province_name = word
        new_fact += word
    return new_fact
```

图 4.3: 地名替换核心代码

4.2.2　基于 VAE 的训练数据生成

基于 VAE 的训练数据生成方法利用生成模型学习数据分布的性质，将生成模型应用在文本数据扩增领域中。变分编码器（VAE）是一种无监督学习复杂数据分布的生成网络模型，其原理为学习可测函数，使任意的概率分布可以映射到待模仿数据的数据分布，因此生成的数据与原有数据具有相似的数据分布，可以作为扩增的训练数据来训练模型。

系统中的 VAE 结构参考了 [39] 中的思想与模型架构，encoder 与 decoder 采用单层的 LSTM 结构来适应文本的时序特征，decoder 以采样自高斯分布的向量 z 为初始状态，参数由 encoder 中的线性层得到，如公式 4.1所示。其中，μ、σ 分

别表示 encoder 网络中产生的均值与方差，r 为在标准高斯分布中采样的随机值。

$$z = \mu + r * e^{\frac{\sigma}{2}} \tag{4.1}$$

利用 VAE 训练文本生成的模型结构如图 4.4所示。VAE 网络在训练过程中的损失函数由两部分组成，一部分负责控制 encoder 网络生成的向量 z 的分布尽可能接近标准高斯函数；另一部分负责控制 decoder 网络生成的文本尽可能与原始文本的分布接近，其损失函数如公式 4.2所示。其中，KL_loss 表示向量 z 与标准高斯函数的 KL 散度值，KL_weight 表示系统设定的 KL 散度所占损失函数的权重，$CrossEntropyloss$ 表示生成的文本与原始文本的交叉熵函数，$input$ 为原始文本，$f(x)$ 为生成文本。

$$Loss = KL_loss(N(\mu, \sigma), N(0, 1)) * KL_weight + CrossEntropyloss(input, f(x)) \tag{4.2}$$

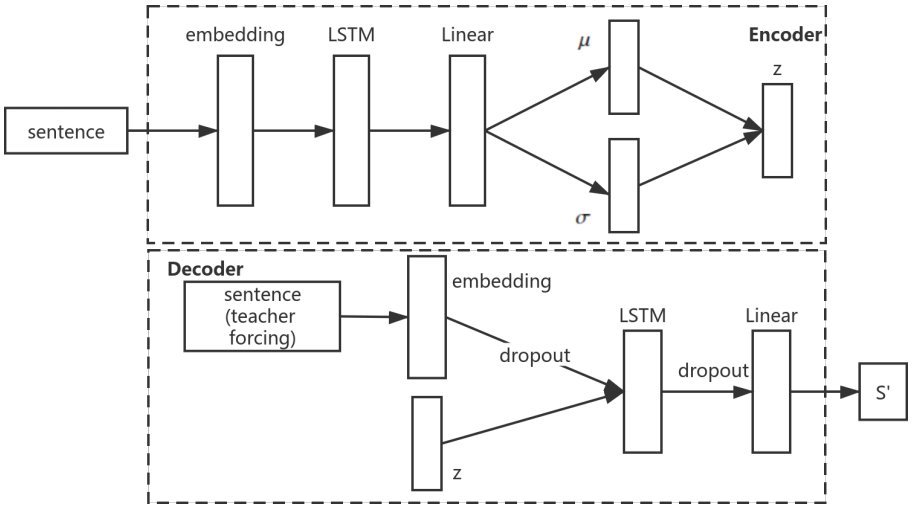


图 4.4: VAE 文本生成模型训练结构图

在系统中基于 VAE 进行数据生成的过程主要分为两部分。第一部分为 VAE 模型的训练过程，用户在输入待生成样本后，系统对数据进行预处理，输入 VAE 结构中进行模型的训练。模型训练过程需要 encoder 网络与 decoder 网络的同时参与，使 encoder 网络的输出值 z 尽可能接近标准正态分布，使 decoder 网络输出值尽可能接近原句。第二部分为 VAE 模型的生成过程，经过训练的 VAE 模型的 decoder 部分已经具备以标准正态分布为输入，模拟生成与原数据分布尽可能

相同的新数据的能力。因此，如图 4.5所示，系统根据用户对生成句子数量参数的要求，在标准正态分布中随机取样作为 decoder 网络的输入值，decoder 网络将产生同样数量的生成文本来模拟原始数据。

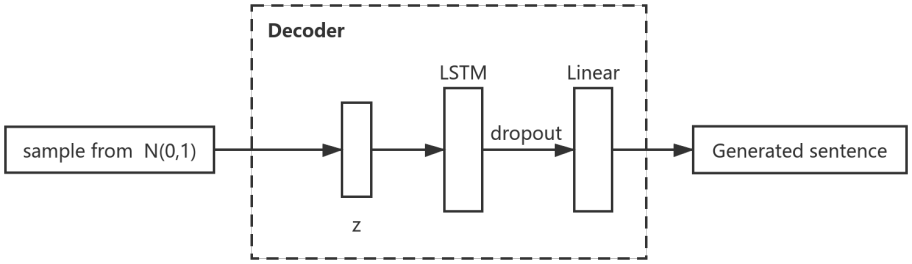


图 4.5: VAE 文本生成模型生成过程结构图

VAE 类负责定义 VAE 模型的网络结构及损失函数的计算等结构性函数。其核心代码如图 4.6所示。Encoder 与 Decoder 类分别定义了 encoder、decoder 网络结构，VAE 类内实现了 Encoder 与 Decoder 类实例。其中，reconstruct 函数用于将 test 值做重构，比对生成值与原始值区别；loss 函数定义了 VAE 网络的损失函数；sample_sentences 函数负责利用随机的标准正态分布做文本生成。

为提高 VAE 模型生成句子的质量，系统对 VAE 生成的句子进行了过滤。首先统计了句子中的词频，如果存在词语，其出现频率大于句子句子长度的 12%，将其去掉；第二步，系统对生成句子进行优化，去掉句子中连续循环的词组，只保留一组该词组；第三步，系统去除了长度小于 80 的无明确意义的句子；最后，对经过以上过滤步骤后的句子进行去重，得到最终的生成句子集合。

系统提供了用户可自定义的训练模型以及生成文本的超参数，并给出默认值，用户可以自定义调节训练过程以及生成文本的数量，具体参数如表 4.3所示。

表 4.3: VAE 模型可调整超参数

参数名称	默认值	说明
max_epoches	100	训练时最大的 epoch 数
kl_start	0.0	KL_weight 的起始值
lr_start	1	learning rate 起始值
warm_up	20	在多少个 epoch 内把 KL weight 恢复到 1
decay_epoch	2	当多少个 epoch loss 不下降时开始更换 learning rate
lr_decay	0.5	learning rate 下降速度

用户在输入的文本中需标记输入本文的对应标签，系统将以标签为单位，将输入数据分为多个训练集，分别进行 VAE 生成模型的训练。最终用户输入的标

```
class VAE(nn.Module):
  def __init__(self, encoder, decoder):
    super(VAE, self).__init__()
    self.encoder = encoder
    self.decoder = decoder
    loc = torch.zeros(encoder.latent_size, device=decoder.device)
    scale = torch.ones(encoder.latent_size, device=decoder.device)
    self.prior = torch.distributions.normal.Normal(loc, scale)
  def reconstruct(self, inp, length): #重构函数
    z, _ = self.encoder.encode(inp, length)
    res = self.decoder.greedy_decode(z)
    return res
  def loss(self, inp, length, kl_weight): #损失函数
    z, KL = self.encoder.encode(inp, length)
    rec_loss = self.decoder.reconstruct_error(inp, length, z)
    return rec_loss + kl_weight * KL, KL * kl_weight
  def sample_sentences(self, nsamples=1): #随机采样生成文本
    z = self.prior.sample((nsamples,))
    res = self.decoder.greedy_decode(z)
    return res
```

图 4.6: VAE 结构核心代码

签数量与 VAE 生成模型的数量相等。用户输入训练文件后，系统将根据用户的输入文件及自定义参数进行 VAE 生成模型的训练，模型训练完成后，将更改 Models 数据表中的模型状态为可用，并将训练完成的模型存储在文件系统中。当用户使用模型进行文本生成时，可以直接调用训练好的模型，避免了模型的多次训练。VAE 模型管理界面如图 4.7所示。

图 4.7: VAE 模型管理界面

4.2.3　含噪声项测试数据生成

含噪声项测试数据生成的目的是为司法领域人工智能系统提供测试模型抗噪能力的测试数据。因此生成的数据对原数据的语法与语义进行了适当修改，与原始数据分布不同。系统中含噪声项测试数据生成方式共三种：含冗余项测试数据生成、含缺失项测试数据生成、含乱序项测试数据生成。

含冗余项测试数据生成的具体实现为：输入文本经过分词后，产生 n 个词语。含冗余项测试数据生成方法对输入文本按照用户给定的插入冗余项比例 r，从司法文本的常用词语词典中按比例随机选择词语 $r*n$ 个，插入输入文本中。系统内置了在 15 万条司法文书中获取的常用 5000 词词典（去除了停用词），词典的 key 为词语，value 为出现次数。在对输入文本添加冗余项时，以词语出现次数作为衡量词语被选择的概率大小，将词语填入原始文本的 $r*n$ 个随机位置中，其流程图如图 4.8所示。含冗余项测试数据生成的核心代码如图 4.9所示。

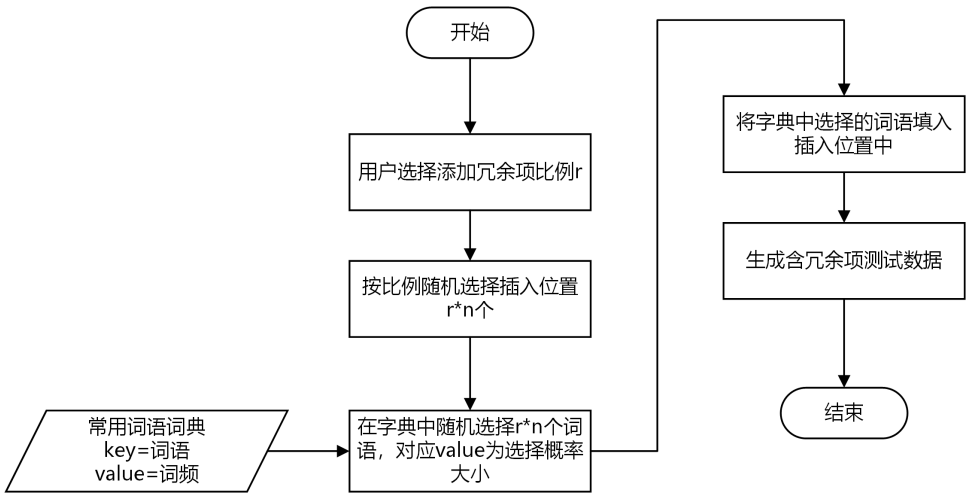


图 4.8: 含冗余项测试数据生成流程图

含缺失项测试数据生成的具体实现为：输入文本经过分词后，产生 n 个词语。含缺失项测试数据生成方法对输入文本按照用户给定的含缺失项比例 r，从 n 个词语中随机选择 $r*n$ 个删除，形成含有一定词语数量缺失的数据。

含乱序项测试数据生成的具体实现为：含乱序项测试数据生成分为词语乱序与句子乱序两部分。输入文本经过分词后，产生 n 个词语，m 个句子。当用户选择词语乱序时，系统根据用户给定的含词语乱序项比例 $r1$，随机选择 $r1*n$ 对词语进行位置互换。当用户选择词语乱序时，系统根据用户给定的含句子乱序项比例 $r2$，以句号作为句子的分割标志，随机选择 $r2*m$ 对句子进行位置互换。

```
def redundancy_gen(self, line_list, param, output_name):
    random_percent = float(param['radio_redun'])
    for line in line_list:
        random_cnt = round(len(line)*random_percent)
        words = random.choices(self.vocabs, weights=weight ,k=random_cnt)
        for j in range(random_cnt):
            loc = random.randrange(0, len(line), 1)
            line.insert(loc, words[i])
        write_str = ''.join(line)
        self.write_json_file(output_name, write_str)
```

图 4.9: 含冗余项测试数据生成核心代码

4.2.4　对抗攻击测试数据生成

对抗攻击测试数据生成的目的是为司法领域人工智能系统提供测试模型抗对抗攻击能力的测试数据。对抗攻击的原理为：对于原始测试数据 X，其在预测模型上得到的预测结果 $Predict(X) = Label_X$，在对原始测试数据做了某些改动后得到 X_{attack}，使其在预测模型上得到的预测结果 $Predict(X_{attack}) \neq Label_X$，且对原始测试数据的改动要尽可能小。在对文本进行对抗攻击时，参考了 [40] 中的定义与算法模型，对测试数据的改动表现为词语的替换，而为了使原始测试数据改动尽可能小则主要表现为两点，一是改动词语的数量尽可能少，二是使改动后词语与原词的语义尽可能相近，也就是说，使两者词向量的距离尽可能小。

为了度量词语之间的距离，系统在 fasttext 中文词向量的基础上，使用 CAIL 司法数据集对词向量进行再次训练，增加其对于司法领域相关词汇的理解，构造适用于司法领域的词向量。为了增加对抗攻击时司法领域常用词的比重，并减少中文词向量中不常见词语，系统选择了词向量中的 67000 个词，其中包括 fasttext 中文词向量的前 50000 个词（fasttext 词向量按词语出现频率排列），以及在 CAIL 司法数据集中出现超过 20 次的词语。

为了减少词向量距离的多次计算造成的内存浪费，系统计算了 67000 个词两两之间的词向量距离，生成词向量距离列表 $dist_mat$，存储了与每个词距离最近的 100 个词语，以及词之间的距离值，$dist_mat$ 大小为 [67000,[100,100]]。系统采用欧几里得距离作为词向量间距离的度量方式，如公式 4.3所示，表示两个词 p, q 的词向量间的距离，词向量维度为 n。

$$diss(p,q) = \sqrt{\sum_{i=1}^{n}(x_i^p - x_i^q)^2} \tag{4.3}$$

系统为了在对原始测试数据改动尽可能小的前提下，生成可以对抗攻击预测模型的测试数据，因此提出了基于遗传算法的对抗攻击测试数据生成方法。遗传算法是一种解决组合优化问题的算法，每一次迭代时，在候选解决方案中根据定义的 fitness 函数来评估这些候选解的质量，以候选解质量决定生成子代的概率。下一代通过父代交叉与变异相结合而产生。由于使用了大量的候选解，因此在对抗攻击中更容易找到成功的对抗攻击文本，使所需修改次数更少。

Algorithm 1: 寻找句子的最佳替换方式

Input: X_{cur}: 待替换的文本
Output: X_{rep}: 经过替换的文本
foreach *word in* X_{cur} **do**
    simwordnum ← get_neighbors(word,dist_list,threshold).Length
idx ← random.choice(X_{cur},p=$Normalize$(simwordnum))
replace_list ← get_neighbors(idx,dist_list,threshold)
X_{rep}_list ← do_replace(X_{cur},replace_list)
X_{rep}_list ← uselm(X_{rep}_list,top_n)
X_{cur}_label, X_{cur}_score ← model.predict(X_{cur})
$X_{rep}_list_score$ ← model.predict_lab(X_{rep}_list,X_{cur}_label)
X_{rep} ← X_{rep}_list[min($X_{rep}_list_score$).index]
if min($X_{rep}_list_score$) < X_{cur}_score **then**
    return X_{rep}
else
    return X_{cur}

在对抗攻击测试数据生成中，首先需要定义在句子中挑选词语并选择最佳词语进行替换的规则。替换算法如算法 1所示。该算法输入为待替换的句子 X_{cur}，算法在该句子中随机选择要替换的词语 idx，然后在与词语 idx 的距离在 $threshold$ 内的临近词中遍历，选择合适的替代词 rep，既与 idx 意义相近，又可以符合该位置的语法环境。假设待替换句子 X_{cur} 被预测模型识别的标签 $label = A$，其置信度为 $score^A_{cur}$，因此需要使替换后的句子对于标签 A 的置信度 $score^A_{rep} < score^A_{cur}$，并在所有合适的替代词中选择对于标签 A 置信度最低的。若不存在符合该规则的替代词，则返回原句 X_{cur}。

算法在输入的句子中随机选择替换词语 idx 时，概率权重为句子中每个词所拥有的距离在 $threshold$ 内的临近词的数量。在对 idx 进行替换时，选择与 idx

意义相近的词语的具体操作为在词向量距离列表 *dist_mat* 中选择与 *idx* 间距离在 *threshold* 内的词语。是否符合替换词语位置语法环境的判定通过语言模型来实现。本文使用 BERT 在中文词语上预训练的语言模型，根据语言模型得分对替换候选词进行排序，保留得分最高的 *top_n* 个候选词。

基于遗传算法进行对抗攻击测试数据生成的优化函数如算法 2所示。算法首先通过调用算法 1, 对原始句子 X_{ori} 进行 S 次替换，生成 S 个初代修改后的句子 POP^0。然后，计算当前一代句子中每个句子在适应度函数上的得分 F。

Algorithm 2: 生成对抗攻击样本

```
for i = 1,...,S in population do
    POP_i^0 ← Algorithm1(X_ori)
for g = 1,...,G in generations do
    for i = 1,...,S in population do
        F_i^{g-1} = f(POP_i^{g-1})
    X_adv = POP_{argmax_j F_j^{g-1}}
    if predict.label(X_adv) ≠ predict.label(X_ori) then
        return X_adv ▷ {对抗攻击成功}
    else
        POP_1^g = {X_adv}
        p = Normalize(F^{g-1})
        for i=2,...,S in population do
            在 POP^{g-1} 中以 p 为概率采样得到 parent_1
            在 POP^{g-1} 中以 p 为概率采样得到 parent_2
            child ← Crossover(parent_1, parent_2)
            child_mut ← Algorithm1(child)
            POP_i^g = {child_mut}
```

算法将适应度函数 $f(X_{rep})$ 的定义如公式 4.4所示。$predict.label(X_{ori})$ 为原始句子的预测标签，$f(X_{rep})$ 表示经过替换的句子 X_{rep} 在原始句子的预测标签上置信度的反比。也就是说，置信度越高，适应性函数得分越低。

$$f(X_{rep}) = 1/predict.score(X_{rep})[predict.label(X_{ori})] \tag{4.4}$$

在当前一代句子中选择适应度函数得分最高的句子 X_{adv}，如果预测模型对 X_{adv} 的预测标签与对原始句子 X_{ori} 的预测标签不一致，则对抗成功，以此条句子为基础生成的对抗攻击测试数据为 X_{adv}。否则，将从当前一代生成的句子中随机抽取，抽取概率与适应度函数得分成正比。将抽取的句子两两一组经过 *Crossover* 函数进行交叉，从两个句子中随机采样合成新句子，最后使用算法 1生成下一代句子的集合。如果在迭代次数 G 之内未生成可对抗攻击模型的句子，则表示对抗失败。

为了度量输入模型的抗对抗攻击能力，本文确定了量化的抗对抗攻击能力指标。用户输入的测试数据数量为 N，将每一条测试数据 $data_i$ 分别对输入的深度学习进行对抗攻击，基于以上所述的遗传算法改变句子中的词语。若对抗成功，则可得修改的词语数量 $attacknum_i$。测试数据 $data_i$ 中所含的有意义词语个数为 $wordnum_i$，即系统对句子去除停用词后的词语个数。抗对抗攻击能力公式如公式 4.5所示。深度学习模型的抗对抗攻击能力与其在对抗攻击中测试数据平均需修改的词语比例成正比。若存在对抗攻击不成功的测试数据 $data_k$，则定义 $attacknum_k = wordnum_k$。抗对抗攻击能力分数最高值为 1，最低值无限趋近于 0。在度量抗对抗能力时，需要使被度量的深度学习模型在对抗攻击中保持相同参数值。例如遗传算法迭代次数、词向量间属于同义词的最大距离等。

$$score_{attack} = \frac{1}{N}\sum_{i=1}^{N}\frac{attacknum_i}{wordnum_i} \tag{4.5}$$

系统提供了在对抗攻击测试数据生成中用户可自定义的参数，并给出默认值，用户可以自定义调节参数大小，具体参数如表 4.4所示。

表 4.4: 对抗攻击测试数据生成中可调整参数

参数名称	默认值	说明
threshold	3.0	定义词向量间属于同义词的最大距离
generations	1000	遗传算法中的最大迭代次数
population	20	遗传算法中每一代生成的句子数量
top_n	20	语言模型中选择的候选词数量

4.3　数据交互

数据交互模块负责用户与系统间的数据及文本的交互，将用户、应用系统、MySQL 数据库系统、HDFS 文件管理系统进行了连接，其功能包括上传文件、存

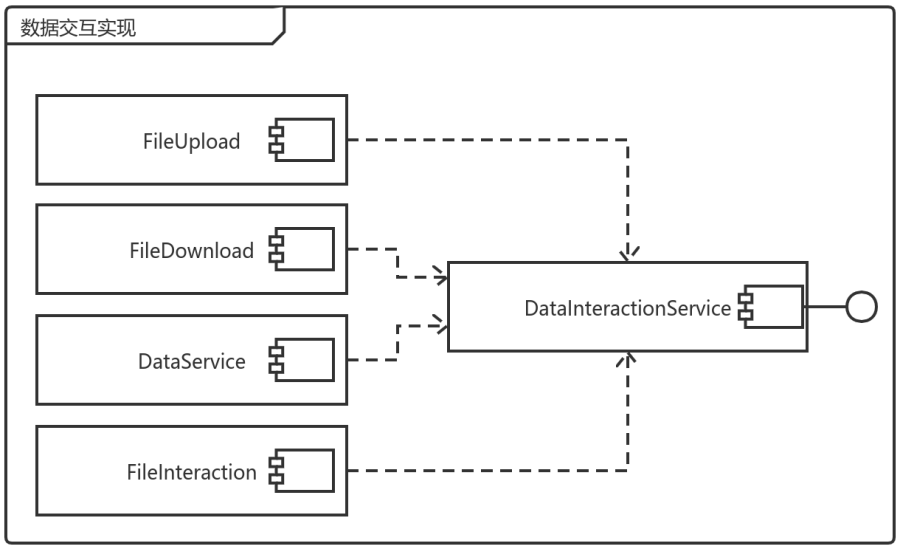


图 4.10: 数据交互实现逻辑

储上传的用户文件、存储及查询用户数据、展示生成数据、下载文件至用户浏览器等。如图 4.10展示了数据交互的实现逻辑。

FileUpload 组件从用户浏览器的 HTTPRequest 对象中获取上传文件，校验上传文件的格式是否与系统要求一致，包括文本文件的数据格式校验与模型文件的上传校验。在文件格式一致的情况下，将文件与文件信息传入 DataService 组件，存储至 HDFS 文件系统中，并将文件信息存储至 MySQL 数据库中。

FileDownload 组件获取待下载文件在数据库存储中的键值，传入 DataService 组件，在 MySQL 数据库中获取文件存储位置，从 HDFS 文件系统中下载文件至用户浏览器。

DataService 组件是 web 传输的数据及文件与 MySQL 数据库系统、HDFS 文件系统的传输介质，负责将用户传输的数据及文件传递给 MySQL 与 HDFS，也负责将用户传入前端的参数传递到业务逻辑代码端。在系统生成结果展示时，DataService 组件负责将生成的 json 文件传递给前端，前端将利用 JavaScript 函数将文本按类别展示在界面中。系统生成结果展示界面示例如图 4.11所示。

FileInteraction 组件为系统与数据库系统、HDFS 文件系统交互的接口。与 HDFS 文件系统的交互包括文件读取、文件写入、文件存储、文件下载等；与 MySQL 数据库的交互包括系统数据写入、数据查询等。

由于系统的功能需要支持大量较大文件的交互，因此在系统中配置了 Hadoop 核心组件环境，利用 Hadoop 核心组件中的 HDFS 文件管理系统，支持用户文件

图 4.11: 系统生成结果展示界面示例

的分布式存储，为系统提供可拓展的文件存储空间，以及防止用户数据的丢失。

系统采用 HDFS 主从架构，架构由三部分角色组成：NameNode 用于与 HDFS client 的读写请求交互，为 client 提供文件位置信息，并承担对 Block、副本策略等的管理功能；Secondary NameNode 辅助 NameNode 的一部分管理工作，负责合并文件信息并传送至 NameNode；DataNode 负责执行 NameNode 提供的读写命令，并负责将文件分配存储至 Block 及从多个 Block 读取文件数据。架构图如图 4.12所示。关键配置信息如图 4.13与图 4.14所示。

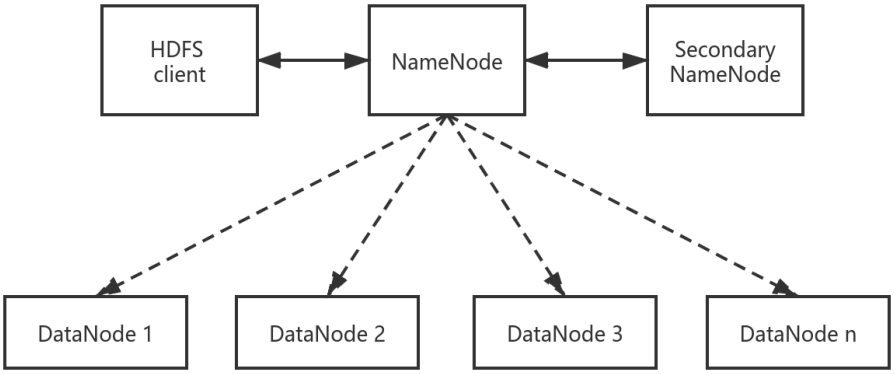


图 4.12: HDFS 设计架构图

4.4　用户文件管理

用户文件管理模块为用户提供了生成文本文件的管理功能。用户文件管理分为四个管理列表，分别为：基于规则的训练数据文件列表、基于 VAE 的训练数据文件列表、含噪声项测试数据文件列表、以及对抗攻击测试数据文件列表。在每个文件列表中，展示了用户历史生成过的所有文件。包括文件名称、文件大

```
<configuration>
    <property>
        <!-- 指定SecondaryNamenode所在地址 -->
        <name>dfs.namenode.secondary.http-address</name>
        <value>Master:50090</value>
    </property>
    <property>
        <name>dfs.replication</name>
        <value>3</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:/usr/local/bigdata/hadoop-2.7.1/tmp/dfs/name</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:/usr/local/bigdata/hadoop-2.7.1/tmp/dfs/data</value>
    </property>
</configuration>
```

图 4.13: hdfs.xml

```
<configuration>
    <!-- 指定HADOOP所使用的文件系统schema（URI）-->
    <property>
        <name>fs.defaultFS</name>
        <value>Master:9000</value>
    </property>

     <!-- 指定hadoop运行时产生文件的存储目录 -->
    <property>
        <name>hadoop.tmp.dir</name>
        <value>file:/uer/local/bigdata/hadoop-2.7.1/tmp</value>
        <description>Abase for other temporary directories.</description>
    </property>
</configuration>
```

图 4.14: core.xml

小、文件生成时间等基本信息。当用户打开个人中心界面时，系统在数据库中查询用户生成过的所有文件信息，按照文件类别显示在文件列表中。

用户文件管理提供了文件下载及文件删除功能。文件下载功能的作用是为用户提供文件下载接口，支持文件的重复多次下载，避免相同文件的多次生成。而且在基于 VAE 的训练数据生成过程中，数据的生成需经过 VAE 模型的训练，耗时较长。用户文件管理可以支持数据的离线生成，用户只需要在数据生成结束后登陆个人中心进行生成文件的下载。若用户选择删除文件，系统将首先在数据库中查询文件在 HDFS 系统中的存储位置，接着删除 MySQL 数据库中的文件信息，并删除 HDFS 文件系统中的存储文件，可以避免造成文件系统中无用文件的大量存储。如图 4.15为用户文件管理的系统界面。

图 4.15: 用户文件管理系统界面

4.5　案例及实验分析

本节给出了司法文本数据自动化生成系统的具体应用案例，并对生成数据的可用性做出了实验分析。本文选取 2018 法研杯 CAIL 数据集 [41] 中的文本数据作为案例分析的输入司法文本。该数据集来源于“中国裁判文书网”，并经过标签提取，每份数据样本包括案情描述、罪名、案例相关法条、判处罚金等字段。其中，案情描述字段为文本格式，表述了案件发生的具体情况，并隐去了罪名等判决信息，每条案情描述包含几个句子，长度在 100-300 字左右，该数据集可支持深度学习预测模型的训练，以根据案情事实预测罪名、罚金等判决结果。

4.5.1　训练数据生成案例及实验分析

本文在 CAIL 数据集中选取数据样本，输入基于规则的训练数据生成方法中。每条样本以案情描述作为输入文本，以案件所属罪名名称作为标签。选择 Blind 扩增方式、反事实扩增方式、同义替换扩增方式，生成文本示例如表 4.5所示。基于规则的训练数据生成方法不改变原始文本语义，可以生成语法正确的、语义完整的新文本。

本文在 CAIL 数据集中分别选取了罪名为盗窃；走私、贩卖、运输、制造毒品；故意伤害；交通肇事的四种案情事实文本共计 28815 条。分别作为 VAE 生

表 4.5: 基于规则的训练数据生成文本示例

原始文本	陕西省咸阳市礼泉县人民检察院指控：2003 年 3 月 2 日 19 时许，被告人王某在江桥街道饭店找李某辩论事项时，双方发生争斗，导致李某脸部、耳朵部位划伤。
Blind 扩增	**Province** 省 **City** 市 **County** 人民检察院指控：**Year** 年 **Month** 月 **Day** 日 **Time** 时许，被告人 **Name1** 在江桥街道饭店找 **Name2** 辩论事项时，双方发生争斗，导致 **Name2** 脸部、耳朵部位划伤。
反事实扩增	**山西省吕梁市文水县**人民检察院指控：**2011 年 5 月 23 日 10 时许**，被告人**孙某**在江桥街道饭店找**赵某**辩论事项时，双方发生争斗，导致**赵某**脸部、耳朵部位划伤。
同义替换扩增	陕西省咸阳市礼泉县人民检察院指控：2003 年 3 月 2 日 19 时许，被告人王某在江桥街道**饭馆**找李某**讨论事宜**时，双方发生**争端**，**致使**李某脸部、耳朵部位划伤。

成模型的训练集、验证集与测试集，在系统上进行基于 VAE 的数据生成实验。数据详细信息如表 4.6所示。由于同一条案情描述可能具有多条罪名标签，因此在处理输入数据时，将具有多条标签的训练数据分别输入在其具有的所有标签类别中。

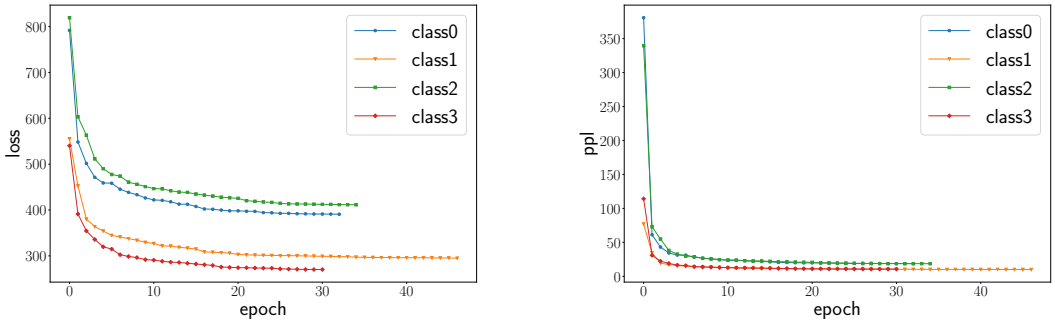
表 4.6: VAE 生成模型的输入数据详情表

标签	标签名称	训练集数量	验证集数量	测试集数量
class0	盗窃	6977	746	1515
class1	走私、贩卖、运输、制造毒品	7523	736	1429
class2	故意伤害	4693	555	1028
class3	交通肇事	2822	278	513

四组训练数据在系统中经过训练，分别学习其类别中文本的分布特征，得到四个 VAE 生成模型。四个 VAE 生成模型训练过程中验证集上的 *Loss* 值、*PPL*（困惑度）值如图 4.16所示。其中，*PPL*（perplexity）[42] 用于表示在每个时间步上，等可能性的输出结果的个数。*PPL* 越小，表示等可能性输出结果个数越少，生成模型越好。其公式表示如公式 4.6所示。其中，ε_{test} 表示测试集的语料库，N 为测试集文本数量。根据图 4.16可以看出，四个 VAE 生成模型均在最大 epoch 中得到较好收敛，且 *PPL* 值稳定在 0-20 之间，具有较强的生成能力。

$$PPL(\varepsilon_{test}) = \exp\left(Loss \cdot N / length(\varepsilon_{test})\right) \tag{4.6}$$

生成的文本范例如表 4.7所示，生成的文本语句通顺，具有明确语意。在 VAE 模型训练过程中，系统将输入文本进行预处理时，隐去了数字及英文字母信息，



(a) Loss 值变化图　　(b) PPL 值变化图

图 4.16: VAE 生成模型训练过程中 Loss 值与 PPL 值变化图

用特定符号代替，目的是减少数字、英文字母为生成模型带来的特征复杂度。因此生成的文本中使用 __NUM__ 代表数字；__YEAR__，__MONTH__，__DAY__ 分别代表年月日的取值；__ENG__ 代表英文字母；__UNK__ 代表训练集语料库中未出现过的词语。

表 4.7: VAE 生成文本范例

标签	生成范例
class0	公诉机关指控 __YEAR__ 年 __MONTH__ 月 __DAY__ 日 __NUM__ 时许被告人李某在本市江汉区 __NUM__ 号 __NUM__ 室趁被害人张某不备之机窃得被害人李某放在床上的苹果手机 __NUM__ 部价值人民币 __NUM__ 元。__YEAR__ 年 __MONTH__ 月 __DAY__ 日被告人李某被公安机关抓获。
class1	公诉机关指控 __YEAR__ 年 __MONTH__ 月 __DAY__ 日 __NUM__ 时许被告人李某在本市海曙区 __UNK__ 路 __NUM__ 号 __NUM__ 栋 __NUM__ 单元 __NUM__ 室其家中以人民币 __NUM__ 元的价格将克甲基苯丙胺贩卖给吸毒人员李某。上述事实有被告人张某某的供述及户籍证明证人证言公安机关出具的抓获经过毒品扣押清单、称量记录及检验报告劳教决定书等证据证实足以认定。
class2	公诉机关指控 __YEAR__ 年 __MONTH__ 月 __DAY__ 日 __NUM__ 时许被告人张 __NUM__ 某在本市海淀区因琐事与被害人刘某发生口角后被告人张 __NUM__ 某用拳头将被害人刘某打伤。经鉴定被害人刘某的损伤程度为轻伤二级。公诉机关认为被告人王 __NUM__ 某的行为已构成提请本院依法判处。
class3	公诉机关指控 __YEAR__ 年 __MONTH__ 月 __DAY__ 日 __NUM__ 时许被告人王某甲酒后驾驶苏 __ENG__ 号小型轿车沿本市河北区 __UNK__ 路由北向南行驶至 __UNK__ 路口时与前方同向行驶的由被害人王某驾驶的苏 __ENG__ 号小型轿车发生碰撞造成两车损坏的交通事故。经鉴定被告人张某甲血液中乙醇含量为 __ENG__ 。经交警部门认定被告人王某甲承担事故的全部责任。案发后被告人张某甲与被害人达成赔偿协议。

在基于规则与基于 VAE 的两种训练数据生成方式下，司法文本数据自动化生成系统将用户输入的训练数据进行扩增，生成与输入的训练数据具有相同分布的新数据。为了验证系统生成训练数据的可用性，即通过系统的训练数据生

成，是否可以使深度学习模型的训练效果得到提升，本文设计了以下实验。

本文随机选取了 1 万条 CAIL 数据集中的训练数据，作为实验的原始训练数据。训练数据的输入值为案情事实文本，标签为案情所属罪名。将 1 万条训练数据传入司法文本数据自动化生成系统，生成经过扩增的训练数据集共 4 万条，包括利用 blind 扩增方式生成的 1 万条文本、反事实扩增方式生成的 1 万条文本、同义词替换方式生成的 1 万条文本、基于 VAE 生成方式生成的 1 万条文本。

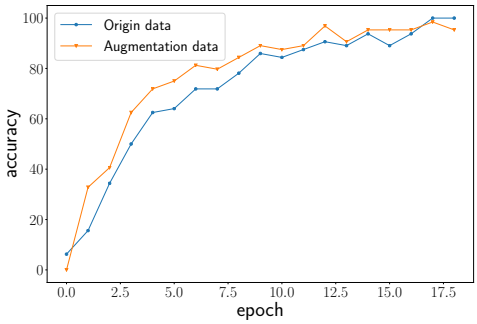
实验中分为四个训练集，利用每个训练集分别训练罪名预测模型。其中，训练集 1 为 1 万条原始数据；训练集 2 在原始数据中加入 3 万条基于规则扩增得到的数据；训练集 3 在原始数据中加入 1 万条基于 VAE 扩增得到的数据；训练集 4 在原始数据中加入基于规则与基于 VAE 的数据，共 5 万条。实验保证各模型在训练过程中的训练次数一致。其中，模型测试集数据为 CAIL 数据集中的原始测试数据，损失函数为交叉熵损失函数值，准确率为预测正确样本与预测错误样本的比值。

实验中模型的网络结构分别为 fastText [43]、TextCNN [44]、LSTM [45] 结构。将四组训练集分别输入三种网络结构，训练罪名预测模型，对比其训练过程中的表现差异。训练过程中各组数据的表现情况如表 4.8所示。根据表格信息可以得出，在加入本文提出的两种数据扩增数据后，三个网络结构的预测准确率均得到较大提升。

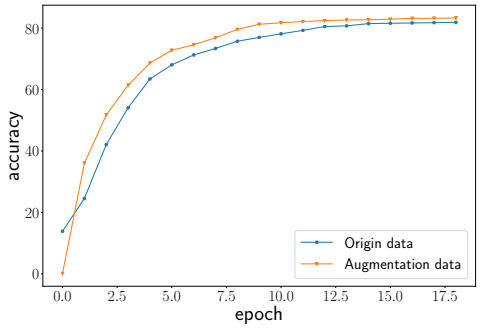
表 4.8: 加入扩增数据前后模型预测准确率对比表

	FastText	TextCNN	LSTM
训练集 1（原始数据）	87.69%	81.91%	71.47%
训练集 2（加入基于规则数据）	87.93%	82.35%	75.71%
训练集 3（加入基于 VAE 数据）	87.88%	82.19%	72.54%
训练集 4（加入基于规则与 VAE 数据）	**88.21%**	**83.31%**	**78.21%**

图 4.17展示了利用原始数据与扩增后数据在 TextCNN 模型进行训练的过程，（a）展示了模型在训练集上的准确率值变化情况，（b）展示了模型在测试集上的准确率值变化情况。原始数据在训练集上的准确率已达到 100%，测试集上的准确率与其具有较大差异，在测试集上的泛化能力较差，模型存在过拟合现象。扩增数据集的加入降低了模型在训练集上的准确率，而测试准确率得到了提升，缓解了模型的过拟合程度，使模型的泛化能力得到提升。可见系统提供的司法文本训练数据的生成方法是可行有效的，扩增数据的加入可以提高模型泛化能力，提高模型的测试准确率。



(a) 训练集准确率值变化图



(b) 测试集准确率值变化图

图 4.17: TextCNN 预测模型训练过程中准确率值变化图

4.5.2　测试数据生成案例及实验分析

本文在 CAIL 数据集中选取数据样本，输入含噪声项测试数据生成方法中。每条样本以案情描述作为文本，以案件所属罪名名称作为标签。选择生成含冗余项、含缺失项、含乱序项测试数据，并调整含噪声比例为 10%，生成的文本示例如表 4.9所示。

表 4.9: 含噪声项测试数据生成文本示例

原始文本	陕西省咸阳市礼泉县人民检察院指控：2003 年 3 月 2 日 19 时许，被告人王某在江桥街道饭店找李某辩论事项时，双方发生争斗，导致李某脸部、耳朵部位划伤。
含冗余项测试数据	陕西省咸阳市鉴定礼泉县人民检察院指控：2003 年认定 3 月 2 日 19 时许，被告人王某在时许江桥街道饭店找李某辩论事项时，双方发生争斗，导致指控李某脸部、耳朵部位划伤。
含缺失项测试数据	陕西省咸阳市礼泉县人民检察院：2003 年 3 月 219 时许，被告人王某在江桥饭店找李某辩论事项时，双方发生争斗，导致脸部、耳朵部位划伤。
含乱序项测试数据	陕西省咸阳市被告人人民检察院指控年 2003：3 月 2 日 19 时许，礼泉县王某在江桥街道饭店找李某辩论事项时，双方发生争斗，部位李某脸部、导致耳朵划伤。

司法领域的一些深度学习应用场景下，待预测的文本存在噪声，数据质量较低。含噪声项测试数据用于评估司法深度学习模型在预测含噪声的文本时，是否可以保持预测准确率的鲁棒性。本文设计了以下实验，用于比较不同深度学习模型的抗噪能力。实验从 CAIL 数据集中随机选取 1 万条样本，以罪名作为预测标签。分别设置两组训练数据：其中一组将 1 万条样本加入噪声项，生成 1 万条含噪声项样本，组成 2 万条样本的训练集，作为实验组；另一组将 1 万条样本复制一次，组成 2 万条样本的训练集，作为对照组。将两组训练数据集分别输入 TextCNN 网络结构中，训练罪名预测模型。训练完成后，将实验组与对照组

预测模型分别使用原始测试集、含噪声项测试集进行测试，两组预测模型在不同测试集上的表现如表 4.10所示。

表 4.10: 原始测试集与含噪声项测试集实验对比表

	实验组预测模型	对照组预测模型
原始测试集测试准确率	81.212%	81.217%
含噪声项测试集测试准确率	79.671%	81.185%
准确率差值	1.541%	0.032%

根据以上数据可以观察到，实验组与对照组在原始测试集中，模型预测准确率几乎没有差别。但两者在含噪声项测试集上的表现具有较大差异。由于实验组在训练中加入了含噪声项的训练数据，模型学习了含噪声数据的分布特征，因此在含噪声项测试集上的预测准确率与原始测试集差值不大；对照组在训练中未加入噪声数据，因此其在含噪声测试数据上与原始测试数据上的预测准确率具有较大差异。

根据以上实验可以得出结论，在原始测试集上表现相似的深度学习预测模型，在含噪声项测试数据上的准确率可能具有较大差异。因此，含噪声项测试数据可以用于评估司法深度学习模型的抗噪性，作为司法深度学习模型的测试指标之一。当模型在预测准确率上表现相似时，抗噪性指标可以为模型提供另一个维度的评估方向。

本文对对抗攻击测试数据生成方法的可用性做了以下的案例分析。案例中以罪名预测模型作为输入的攻击模型，模型采用 FastText 及 TextCNN 网络结构，在 CAIL 数据集中采样样本作为训练数据，标签为案情描述相应罪名，训练数据中共有 20 个标签，模型属于多分类多标签预测模型。在系统中输入待攻击模型、测试数据及训练集词表，各参数选择为默认值。

表 4.11: 对抗攻击模型信息及对抗攻击情况

	模型网络结构	模型准确率	测试数据数量	对抗攻击成功个数	对抗攻击成功率	文本平均修改比例	抗对抗攻击分数
模型 1	FastText	73.79%	1156	1148	99.30%	0.147	0.153
模型 2	FastText	81.19%	1156	1139	98.52%	0.189	0.201
模型 3	TextCNN	73.42%	1156	1154	99.82%	0.126	0.128
模型 4	TextCNN	86.14%	1156	1113	96.28%	0.243	0.271

表 4.11列举了各对抗攻击模型的模型信息及对抗攻击情况。包括模型结构，模型准确率，对各模型进行对抗攻击时，对抗攻击成功的测试数据所占比例，以及对抗攻击成功文本的平均修改比例。根据以上数据可以计算得出各模型在默

认参数下的抗对抗攻击能力分数。可以看出，模型准确率以及模型结构等因素都会对模型的抗对抗攻击能力产生影响。

表 4.12: 对抗攻击测试数据生成文本示例

	模型预测标签	文本内容
原始文本	盗窃	公诉机关指控，2016 年 5 月 20 日上午，被告人丁某伙同“小某”（另案处理）至本市阳明街道北郊村柏字地东区 *** 号租房，采用撬门等手段进入被害人李某租房内，窃得价值人民币 1700 元的台式组装电脑 1 台。当日晚上，被告人丁某被本市公安民警抓获，后其在取保候审期间脱逃，于同年 10 月 10 日被贵州省纳雍县公安局民警再次抓获。案发后，上述赃物已追回并发还给了被害人。
对抗攻击后文本	制造、贩卖、传播淫秽物品	公诉机关指控，2016 年 5 月 20 日上午，被告人丁某伙同“小某”（另案处理）至本市阳明**派出所**北郊村柏字地**石岐区** *** 号**出租屋**，采用撬门等**有效性**进入**当事人**李某**出租屋**内，窃得**相关性**人民币 1700 元的台式组装**游戏机** 1 台。当日**白天**，被告人丁某被本市公安民警**查获**，后其在取保候审期间脱逃，于同年 10 月 10 日被贵州省**黔东南州**公安局民警再次**查获**。案发后，上述赃物已追回并发还给了**当事人**。

系统展示了攻击前的测试数据及预测标签，以及生成的对抗攻击测试数据的文本及标签。表 4.12举例展示了经过对抗攻击的文本。根据以上案例分析可以说明，系统提供的对抗攻击测试数据生成方法具有较高的对抗攻击成功率，对抗攻击后的文本改变词语数较低且改变后词语对语义影响较小，可以得出合理的抗对抗攻击分数，因此可以作为测试数据应用于司法深度学习模型对抗攻击能力的评估。

4.6　系统测试

本章上述小节给出了司法文本自动生成系统的具体设计及实现。本节对第三章中描述的系统功能分别设计了测试用例，检测系统功能的可用性及完整性。

4.6.1　测试环境

系统测试环境如表 4.13所示。系统服务与集群主节点部署在同一台 Linux 服务器上，集群从节点负责提供 Hadoop 的 DataNode。用户通过 Chrome 浏览器对系统进行访问。

4.6.2　功能测试

本节模拟用户使用流程，对需求分析部分提出的用例进行功能测试，包括数据交互模块、文本预处理模块、数据生成模块以及用户文件管理模块。

表 4.13: 系统测试环境

设备名称	参数	参数详情
用户计算机	操作系统	Windows 10
	Chrome	72.0(64 位)
主节点	操作系统	Ubuntu 16.04.6
	内存	8GB
	Python	3.7.3
	MySQL	5.6
	Hadoop	2.7.1
从节点	操作系统	Ubuntu16.04.6
	内存	8GB
	Hadoop	2.7.1

表 4.14: 数据上传测试用例

用例编号	DQ_RS_01
测试目标	用户可上传数据生成所需文件，系统检测传入文件格式，如文件符合格式，传入系统进行处理并执行生成任务
前置条件	用户登录系统
正常流程	1. 用户按照系统要求的格式整理待上传文件 2. 调用系统的数据上传接口，上传文件 3. 系统接收上传的文件，检测文件格式 4. 如文件格式不符，提示用户重新上传 5. 如文件格式符合要求，文件传入 HDFS 文件系统中，文本文件传入预处理模块进行数据格式化，模型文件传入数据生成模块支持数据生成
预期结果	用户上传文件后，成功进行格式检测并返回结果，如文件符合格式，传入 HDFS 文件系统
实际结果	与预期结果相符

在数据交互模块，主要包含数据上传及文件下载两个测试用例。数据上传测试用例如表 4.14所示，针对系统的数据上传功能进行测试。上传数据生成所需文件，包括文本文件、模型文件等，系统将传入的数据处理后进行数据生成，获取数据生成结果文件。结果表明，测试结果与预期结果相符。

文件下载测试用例如表 4.15所示，针对系统的文件下载功能进行测试。模拟用户操作流程，在数据生成流程执行完成后，提取生成的 json 文件中的相关内容，系统将生成结果写入结果生成页面，并提供文件下载接口，支持用户下载文件至本地。结果表明，测试结果与预期结果相符。

在数据文本预处理模块中，文本预处理测试用例如表 4.16所示，针对系统的文本预处理功能进行测试。在待生成文本文件传入系统后，对文件进行文本

表 4.15: 文件下载测试用例

用例编号	DQ_RS_02
测试目标	在数据生成流程执行完成后，系统将生成结果展示在系统页面，并提供文件下载接口
前置条件	数据生成流程执行完成
正常流程	1. 数据生成流程执行完成，生成结果写入 json 文件中 2. 将 json 文件存储在 HDFS 文件系统中，并将文件信息写入数据库中，将文件键值传递到结果生成页面 3. 系统根据文件键值找到 json 文件，读入 json 文件，写入结果展示页面 4. 在结果展示页面提供文件下载接口，接口根据文件键值在数据库中寻找文件存储地址，下载至用户本地
预期结果	数据生成流程执行完成后，生成结果写入结果生成页面，并提供文件下载接口，用于下载文件至本地
实际结果	与预期结果相符

预处理，包括文本结构化、分词处理、以及在首次进行需要建立字典的生成方式时，对文本关于字典的映射。生成结构化的 json 文件，用于数据生成服务。结果表明，测试结果与预期结果相符。

表 4.16: 文本预处理测试用例

用例编号	DQ_RS_03
测试目标	对用户传入的待生成文本文件进行文本预处理，生成结构化 json 文件
前置条件	待生成文本文件传入系统
正常流程	1. 在 HDFS 文件系统中读取文本文件 2. 对文本文件进行结构化操作生成 json 文件 3. 对文本进行分词处理并写入 json 文件中 4. 首次进行需要建立字典的生成方式时，对文本根据建立的字典映射为 index 值，写入 json 文件 5. 将生成的 json 文件存储在 HDFS 文件系统，为数据生成流程提供数据
预期结果	待生成文本文件传入后，对文本进行预处理，形成结构化文件并存储在 HDFS 系统中
实际结果	与预期结果相符

在数据生成模块中，主要包含基于规则的训练数据生成、基于 VAE 的训练数据生成、含噪声项测试数据生成、以及对抗攻击测试数据生成四个测试用例。

基于规则的训练数据生成测试用例如表 4.17所示，针对系统基于规则的训练数据生成功能进行测试。模拟用户操作，在输入待生成文件后，选择自定义参数，基于规则的生成算法根据用户自定义参数，对文件进行基于规则的数据扩增，并将生成文本存储在 json 文件中。结果表明，测试结果与预期结果相符。

表 4.17: 基于规则的训练数据生成测试用例

用例编号	DQ_RS_04
测试目标	根据用户传入的待生成文件及自定义参数，生成基于规则的训练数据
前置条件	待生成文本文件传入系统并经过预处理，用户选择参数传入系统
正常流程	1. 读取经过预处理的待生成文本文件 2. 将用户自定义参数传入生成算法 3. 基于规则的生成算法根据用户自定义参数，对输入文件进行基于规则的数据扩增 4. 将生成的文本存储在 json 文件中，提交至数据交互模块，保存在 HDFS 文件系统，并向系统返回文件在数据库中存储的键值
预期结果	系统根据传入的待生成文件及自定义参数，生成基于规则的训练数据
实际结果	与预期结果相符

基于 VAE 的训练数据生成测试用例如表 4.18所示，针对系统基于 VAE 的训练数据生成功能进行测试。模拟用户操作，在输入待生成文件后，选择自定义参数传入系统，系统将用户自定义参数作为模型训练的超参数，以输入文件作为 VAE 模型的训练数据，训练 VAE 生成模型，利用训练好的 VAE 模型进行相似文本的生成，将生成的文本存储在 json 文件中。结果表明，测试结果与预期结果相符。

表 4.18: 基于 VAE 的训练数据生成测试用例

用例编号	DQ_RS_05
测试目标	根据用户传入的待生成文件及自定义参数，生成基于 VAE 的训练数据
前置条件	待生成文本文件传入系统并经过预处理，用户选择参数传入系统
正常流程	1. 读取经过预处理的待生成文本文件 2. 将用户自定义参数传入 VAE 模型，作为模型训练的超参数 3. 基于 VAE 的生成算法以输入文件作为 VAE 模型的训练数据，训练 VAE 生成模型 4. 根据用户给定的生成文本数量，利用经过训练的 VAE 模型，生成扩增后的训练数据 5. 将生成的文本存储在 json 文件中，提交至数据交互模块，保存在 HDFS 文件系统，并向系统返回文件在数据库中存储的键值
预期结果	系统根据传入的待生成文件及自定义参数，训练 VAE 生成模型，生成基于 VAE 的训练数据
实际结果	与预期结果相符

含噪声项测试数据生成测试用例如表 4.19所示，针对系统含噪声项测试数据生成功能进行测试。模拟用户操作，在输入待生成文件后，选择自定义参数，含噪声项测试数据生成算法根据用户自定义参数，对输入文件进行含噪声项测

试数据生成，并将生成的文本存储在 json 文件中。结果表明，测试结果与预期结果相符。

表 4.19: 含噪声项测试数据生成测试用例

用例编号	DQ_RS_06
测试目标	根据用户传入的待生成文件及自定义参数，生成含噪声项测试数据
前置条件	待生成文本文件传入系统并经过预处理，用户选择参数传入系统
正常流程	1. 读取经过预处理的待生成文本文件 2. 将用户自定义参数传入生成算法 3. 含噪声项测试数据生成算法根据用户自定义参数，对输入文件进行含噪声项测试数据生成 4. 将生成的文本存储在 json 文件中，提交至数据交互模块，保存在 HDFS 文件系统，并向系统返回文件在数据库中存储的键值
预期结果	系统根据传入的待生成文件及自定义参数，生成含噪声项测试数据
实际结果	与预期结果相符

对抗攻击测试数据生成测试用例如表 4.20所示，针对系统对抗攻击测试数据生成功能进行测试。模拟用户操作，输入生成所需文件，包括预测模型文件、训练集词表、原始测试数据，选择自定义参数，对抗攻击测试数据生成算法根据用户自定义参数，对输入的预测模型进行对抗攻击，生成对抗攻击测试数据，并将生成的文本存储在 json 文件中。结果表明，测试结果与预期结果相符。

表 4.20: 对抗攻击测试数据生成测试用例

用例编号	DQ_RS_07
测试目标	根据用户传入的文件及自定义参数，生成对抗攻击测试数据
前置条件	待生成文本文件传入系统并经过预处理，所需的其他文件传入系统，用户选择参数传入系统
正常流程	1. 读取经过预处理的待生成文本文件 2. 读取输入的预测模型文件、训练集词表及用户自定义参数传入生成算法 3. 对抗攻击测试数据生成算法根据用户自定义参数，对输入的预测模型进行对抗攻击，生成对抗攻击测试数据 4. 将生成的文本存储在 json 文件中，提交至数据交互模块，保存在 HDFS 文件系统，并向系统返回文件在数据库中存储的键值
预期结果	系统根据传入的文件及自定义参数，生成对抗攻击测试数据
实际结果	与预期结果相符

在用户文件管理模块中，用户文件管理测试用例如表 4.21所示，针对系统的用户文件管理功能进行测试。模拟用户操作，在登录系统并生成文件后，进入个人中心，系统将用户历史生成过的文件信息分类展示在列表中，并提供文件

下载及文件删除功能。结果表明，测试结果与预期结果相符。

表 4.21: 用户文件管理测试用例

用例编号	DQ_RS_08
测试目标	显示用户历史生成的文件列表，支持下载与删除文件
前置条件	用户登录系统并生成文件
正常流程	1. 数据库搜索用户生成的文件信息，包括文件名、文件大小、文件生成时间等，按生成类别展示在列表中 2. 用户选择下载文件时，根据数据库中文件的存储位置，从 HDFS 文件系统中下载文件至用户本地 3. 用户选择删除文件时，删除数据库中文件信息，并从 HDFS 文件系统中删除文件
预期结果	系统显示文件列表，并可以成功下载或删除文件
实际结果	与预期结果相符

4.6.3 性能测试

司法文本数据自动化生成系统的核心为数据生成模块，因此本节主要对数据生成模块下的各功能进行性能测试，包括基于规则的训练数据生成、基于 VAE 的训练数据生成、含噪声项测试数据生成、对抗攻击测试数据生成。分别向四种生成方式输入待生成文本，根据默认参数进行数据生成，测试结果如表 4.22所示。其中，基于 VAE 的训练数据生成利用了已训练完成的 VAE 模型进行的数据生成操作，因此没有输入文本。

表 4.22: 系统各生成方式性能测试结果

功能	输入文本数量	生成文本数量	耗时
基于规则的训练数据生成	10000	30000	67.20s
基于 VAE 的训练数据生成	-	10000	11.76s
含噪声项测试数据生成	10000	30000	54.12s
对抗攻击测试数据生成	1000	1000	513.11s

4.7 本章小结

本章对系统的各个功能模块进行了具体实现细节的介绍。首先介绍了文本预处理的具体方法，给出了文本结构化的具体实现。然后介绍了数据生成模块中各生成方法的核心实现，其中包括在基于规则的训练数据生成方法中对各个规则细节的描述；在基于 VAE 的训练数据生成方法中，对 VAE 模型的介绍；在含噪声项测试数据生成中，对各个生成方法细节的描述；在对抗攻击测试数据

生成中，对对抗算法核心实现的描述。其次，介绍了数据交互功能、用户文件管理功能的实现逻辑，以及 HDFS 分布式文件系统的搭建中的配置方法。在案例与实验分析部分，对四种生成算法进行了生成案例分析，并对生成数据的可用性进行了实验。最后，对系统的核心功能进行了系统测试。

第五章　总结与展望

5.1　总结

本文基于司法领域深度学习模型在开发对于文本数据的真实需求，设计并实现了集训练数据生成与测试数据生成于一体的司法文本数据自动化生成系统。为司法领域深度学习模型的开发者提供了与原始数据具有相似分布的高质量训练数据，提供了可以评估模型抗噪能力及抗对抗攻击能力的测试数据。

本文为用户提供了两种训练数据生成方法。训练数据生成方法用于为司法深度学习模型提供数据扩增服务，增加高质量的司法文本训练数据，提高系统预测准确率。其中，基于规则的训练数据生成方法结合司法文本特性，提供了 blind 扩增方法、反事实扩增方法、同义词替换法，尽量不改变文本语法与语义，生成了与原有司法数据具有相同数据分布的高数据质量的文本。基于 VAE 的训练数据生成利用变分自编码器（VAE），在无监督方式下学习原始数据的低维表现形式，生成与原始数据具有一些变化的新的数据。经过实验验证，系统提供的两种训练数据生成方式具有有效性，可以使以 FastText、TextCNN、LSTM 为结构的罪名预测系统的准确率得到提升。

本文为用户提供了两种测试数据生成方法。其中，含噪声项测试数据生成提供了含冗余项、含乱序项、含缺失项测试数据生成方法，生成了具有噪声的测试数据，该测试数据在深度学习模型上的准确率可以评估模型的抗噪能力。对抗攻击测试数据生成在测试数据上做尽可能小的文本改动，对预测模型进行对抗攻击，生成对抗攻击测试数据，用于深度学习模型抗对抗攻击能力的评估。利用系统提供的两种测试数据生成方式，均可以对司法领域深度学习模型做出合理的测试。

本文为用户提供了便于操作的 web 应用，将以上的多种生成方法在应用中进行集成。在数据生成中支持用户自定义生成参数，并以文件形式下载生成数据。实现了统一的文件管理，支持生成文本文件的重复下载。使用 HDFS 作为文件管理系统，实现了系统文件存储的可拓展性。

5.2　进一步工作

本文初步实现了司法文本数据自动化生成系统，实现了司法文本训练数据与测试数据的自动化生成等基本功能，目前系统功能较为单一，与用户的交互方式较少。在本文基础上，还有以下方面可以做进一步修改与完善。

系统的生成速度还有待提升。系统在文本生成过程中只利用了服务器单一的计算资源，文本生成速度受限。而若想获得生成速度的提升，需要将集中式计算模式改为分布式计算。后续将对系统中的生成算法进行分布式计算的搭建，提升文本生成速度。

系统的生成方式还有待扩展。本文在训练数据及测试数据的生成中只设计并实现了部分生成方式。在训练数据的生成中，还有一些通用的数据扩增方法可以经过改进应用在司法文本数据中。对于测试数据的生成，可以定义更广泛维度的测试指标，并根据指标需要建立多种类的测试数据。另一方面，应拓展对更多类型的司法文本数据的支持，例如对裁判文书经过解析的 xml 格式的结构化文本生成等。后续将调研及设计更多种类的司法文本数据生成方法。

系统与用户的交互方式有待丰富。系统目前的用户自定义参数较少，用户在数据生成过程中参与度较低，生成流程与生成的数据模式较单一。后续将设计并实现更多用户自定义选项，使数据生成过程更多样化，拓展与用户的交互方式。

本文中每个数据扩增方式对应的最佳扩增倍数还有待研究。对于系统提出的几种数据扩增方式，本文目前未对其最佳扩增倍数进行探究。后续将通过实验对最佳扩增倍数进行计算并反馈给用户。

参考文献

[1] 李飞, 人工智能与司法的裁判及解释, 法律科学 (西北政法大学学报) 36 (05) (2018) 34–43.

[2] 范明志, 网络司法公开：“互联网＋司法”改革的起跑线, 新华月报 (2018) 92–94.

[3] 伍红梅, 以“大数据＋机器学习”为驱动构建刑事案件判案智能预测系统, 人民司法 (10) (2018) 34–40.

[4] Y. Li, T. He, G. Yan, S. Zhang, H. Wang, Using case facts to predict penalty with deep learning, in: International Conference of Pioneering Computer Scientists, Engineers and Educators, Springer, 2019, pp. 610–617.

[5] G. Yan, Y. Li, S. Shen, S. Zhang, J. Liu, Law article prediction based on deep learning, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion, IEEE, 2019, pp. 281–284.

[6] T. He, H. Lian, Z. Qin, Z. Zou, B. Luo, Word embedding based document similarity for the inferring of penalty, in: International Conference on Web Information Systems and Applications, Springer, 2018, pp. 240–251.

[7] 左卫民, 关于法律人工智能在中国运用前景的若干思考, 清华法学 12 (2) (2018) 108–124.

[8] I. M. A. Agastya, A. Setyanto, Classification of indonesian batik using deep learning techniques and data augmentation, 2018, pp. 27 – 31.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

[10] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, 2015, pp. 448–456.

[11] C. Lei, B. Hu, D. Wang, S. Zhang, Z. Chen, A preliminary study on data augmentation of deep learning for image classification, in: Proceedings of the 11th Asia-Pacific Symposium on Internetware, 2019, pp. 1–6.

[12] H. Lee, M. Ra, W. Kim, Nighttime data augmentation using GAN for improving blind-spot detection, IEEE Access 8 (2020) 48049–48059.

[13] A. H. Jha, S. Anand, M. Singh, V. Veeravasarapu, Disentangling factors of variation with cycle-consistent variational auto-encoders, in: European Conference on Computer Vision, Springer, 2018, pp. 829–845.

[14] X. Lu, B. Zheng, A. Velivelli, C. Zhai, Enhancing text categorization with semantic-enriched representation and training data augmentation, Journal of the American Medical Informatics Association 13 (5) (2006) 526–535.

[15] A. W. Yu, D. Dohan, M. Luong, R. Zhao, K. Chen, M. Norouzi, Q. V. Le, Qanet: Combining local convolution with global self-attention for reading comprehension, in: 6th International Conference on Learning Representations, 2018, 2018.

[16] W. Y. Wang, D. Yang, That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 2557–2563.

[17] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, et al., Deepmutation: Mutation testing of deep learning systems, in: 2018 IEEE 29th International Symposium on Software Reliability Engineering, IEEE, 2018, pp. 100–111.

[18] W. Hsu, Y. Zhang, J. Glass, Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation, in: 2017 IEEE Automatic Speech Recognition and Understanding Workshop, 2017, pp. 16–23.

[19] B. Liang, H. Li, M. Su, P. Bian, X. Li, W. Shi, Deep text classification can be fooled, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 4208–4215.

[20] M. Zhang, Z. Deng, W. Che, T. Liu, Combining statistical model and dictionary for domain adaption of chinese word segmentation, Journal of Chinese Information Processing 26 (2) (2012) 8–12.

[21] 黄昌宁, 赵海, 中文分词十年回顾, 中文信息学报 21 (3) (2007) 8–19.

[22] A. Voina, Statistical estimation in hierarchical hidden markov model, Cybernetics and Systems Analysis 50 (6) (2014) 898 – 912.

[23] J. G. Rogers, H. I. Christensen, A conditional random field model for place and object classification, in: 2012 IEEE International Conference on Robotics and Automation, IEEE, 2012, pp. 1766–1772.

[24] Y. Zhang, Support vector machine classification algorithm and its application, in: International Conference on Information Computing and Applications, Springer, 2012, pp. 179–186.

[25] Z. Li, M. Sun, Punctuation as implicit annotations for chinese word segmentation, Computational Linguistics 35 (4) (2009) 505–512.

[26] J. Ramos, et al., Using tf-idf to determine word relevance in document queries, in: International Conference on Machine Learning, Vol. 242, Piscataway, NJ, 2003, pp. 133–142.

[27] T. Renter, A. Borisov, M. De Rijke, Siamese cbow: Optimizing word embeddings for sentence representations, Vol. 2, Berlin, Germany, 2016, pp. 941 – 951.

[28] J. Pennington, R. Socher, C. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1532–1543.

[29] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, T. Mikolov, Learning word vectors for 157 languages, 2019, pp. 3483 – 3487.

[30] Y. Bengio, R. Ducharme, P. Vincent, A neural probabilistic language model, Denver, CO, United states, 2001.

[31] E. Yilmaz, I. D. El-Kahlout, The use of recurrent neural networks language model in turkish-english machine translation, Trabzon, Turkey, 2014, pp. 1247 – 1250.

[32] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2018, pp. 2227–2237.

[33] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, pp. 4171–4186.

[34] D. P. Kingma, M. Welling, Auto-encoding variational bayes, in: 2nd International Conference on Learning Representations, 2014, 2014.

[35] J. Forcier, P. Bissex, W. J. Chun, Python web development with Django, Addison-Wesley Professional, 2008.

[36] D. Whitley, A genetic algorithm tutorial, Statistics and Computing 4 (2) (1994) 65–85.

[37] T. White, Hadoop: The definitive guide, ” O’Reilly Media, Inc.”, 2012.

[38] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The hadoop distributed file system, in: 2010 IEEE 26th symposium on mass storage systems and technologies, IEEE, 2010, pp. 1–10.

[39] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, S. Bengio, Generating sentences from a continuous space, in: Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, Association for Computational Linguistics, Berlin, Germany, 2016, pp. 10–21.

[40] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. B. Srivastava, K.-W. Chang, Generating natural language adversarial examples, Brussels, Belgium, 2020, pp. 2890 – 2896.

[41] C. Xiao, H. Zhong, Z. Guo, C. Tu, Z. Liu, M. Sun, Y. Feng, X. Han, Z. Hu, H. Wang, J. Xu, CAIL2018: A large-scale legal dataset for judgment prediction, CoRR abs/1807.02478.

[42] R. Sennrich, Perplexity minimization for translation model domain adaptation in statistical machine translation, in: Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, 2012, pp. 539–549.

[43] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, T. Mikolov, Fasttext.zip: Compressing text classification models, arXiv preprint arXiv:1612.03651.

[44] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1746–1751.

[45] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder–decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1724–1734.

简历与科研成果

基本情况 张舒，女，汉族，1996 年 7 月出生，山西省吕梁市人。

教育背景

2018.9 ~ 2020.6 南京大学软件学院 硕士

20014.9 ~ 2018.7 大连海事大学信息科学技术学院 本科

这里是读研期间的成果（实例为受理的专利）

1. **S. Zhang**, G. Yan, Y. Li, J. Liu, Evaluation of judicial imprisonment term prediction model based on text mutation, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE, 2019, pp. 62–65.

2. Y. Li, T. He, G. Yan, **S. Zhang**, H. Wang, Using case facts to predict penalty with deep learning, in: International Conference of Pioneering Computer Scientists, Engineers and Educators, Springer, 2019, pp. 610–617.

3. G. Yan, Y. Li, S. Shen, **S. Zhang**, J. Liu, Law article prediction based on deep learning, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE, 2019, pp. 281–284.

4. C. Lei, B. Hu, D. Wang, **S. Zhang**, Z. Chen, A preliminary study on data augmentation of deep learning for image classification, in: Proceedings of the 11th Asia-Pacific Symposium on Internetware, 2019, pp. 1–6.

5. G. Yan, Y. Li, **S. Zhang**, Z. Chen, Data augmentation for deep learning of judgment documents, in: International Conference on Intelligent Science and Big Data Engineering, Springer, 2019, pp. 232–242.

6. 何铁科，黎宇，**张舒**，刘嘉，严格，“一种基于案情事实的深度刑期预测方法”，申请号：201910352428.5，已受理。

致　　谢

在本篇论文完成之际，我要向所有关心我帮助过我的人致以最诚挚的感谢。

首先感谢我的研究生导师陈振宇老师。陈老师让我认识到作为一名学者严谨与勤奋的态度，认识到对生活需要保有的热情与乐观。陈老师在整个研究生阶段给予了我很大的帮助，从初入实验室时帮助我确定研究方向，到科研过程中遇到问题时对我的建议与指引，再到论文写作过程中的悉心指导，我的毕业成果离不开陈老师的辛勤指导。

同时，我要感谢南京大学软件学院智能软件工程实验室 (Intelligent Software Engineering, iSE) 的所有老师与同学。感谢实验室老师们对我的指导与帮助，感谢师兄师姐给予我学习与工作的经验，感谢实验室同学与我一起度过的两年时光。

谢谢我的家人，在生活中给予了我莫大的帮助，在精神上给予我可以依靠的港湾。在疫情居家期间给予了我舒适的学习环境，让我可以专心学业。

最后感谢为本文答辩评审的各位老师们，谢谢诸位的辛勤工作。

版权与原创性说明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：张舒

日期：2020 年 5 月 21 日