技术报告

李子鸣 2024.8.25

1. 背景  
    在现代信息社会，数据的种类繁多，包括个人信息、财务数据、商业机密、医疗记录等。为了有效管理这些数据，组织通常会根据数据的重要性和敏感性，将其分为几个等级。例如，普通数据、敏感数据和高度敏感数据。普通数据可能是公开的信息，敏感数据则可能涉及个人隐私，而高度敏感数据则可能是涉及国家安全或重大商业利益的信息。

数据分类分级的主要目的在于提高数据管理的效率和安全性。通过对数据进行分类，组织能够更清晰地了解各类数据的特性和风险，从而制定更有针对性的安全策略。例如，对于高度敏感的数据，可能需要实施更严格的访问控制和加密措施，而对于普通数据则可以采取相对宽松的管理策略。此外，数据分类分级还可以帮助组织满足法律法规的要求，确保在数据处理和存储过程中遵循相关的合规标准。

实施数据分类分级的过程通常包括数据识别、分类标准的制定、数据标签的赋予以及定期的审查与更新。通过这些步骤，组织能够持续监控数据的状态，及时调整管理策略，以应对不断变化的安全威胁和业务需求。总之，数据分类分级不仅是信息安全管理的重要组成部分，也是提升组织数据治理能力的有效手段。通过科学合理的数据分类分级，组织能够更好地保护信息资产，降低安全风险，实现可持续发展。

然而，在目前的数据分类分级这一问题中，存在着如下的问题：

1. 存在海量的数据标签，人工分类代价巨大。

2. 不同行业或同一行业内部数据标签往往有着大量的重复字段，却未加以利用。

3. 目前依靠计算机进行自动文本识别和打标的方法较为朴素，往往只是通过字符串检测与匹配的方式进行，正确率低，遗漏率高，且可能无法提取出字段的全部特征。

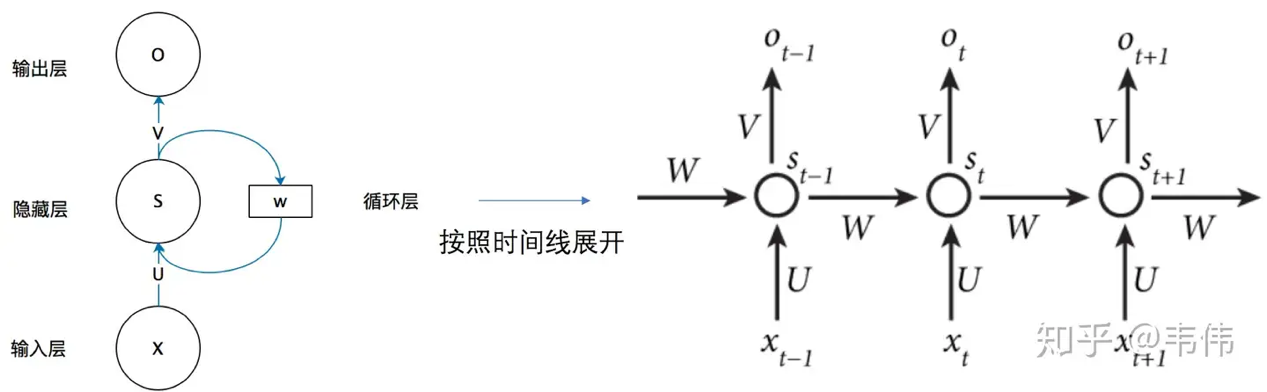
传统的数据分类和打标的算法主要是依靠以下的想法。由于不同行业，数据表的标签可能是非常海量的，因此无法进行人工的数据分类。所以说，为了对于一个新的数据表中的数据进行分类处理，我们一般采用的方法是通过字符串匹配和字符串识别的方法，对于这一数据表中的数据表头进行分类处理。也就是针对这一数据列表中的表头，它一般是一系列的字符串。通过字符串匹配等方式找到之前已分类评定过的表中与其相似的字段，找到的这些与其相似的字段，就可以完成对于数据分类的划分。然而这种方法其效率较低，而且准确率也比较差，亟待改进。同时，这一方法只利用了文本表头的信息，而没有利用文本中内容的信息。

目前，人工智能技术飞速发展，成为下一代引领生产力发展的科技动力，我们注意到，以自然语言处理（NLP）为首的一众机器学习与人工智能的算法可以被利用到处理数据分类分级的任务中来。自然语言处理（Natural Language Processing, NLP）是人工智能领域的一个重要分支，旨在使计算机能够理解、解释和生成自然语言。在数据分类分级的背景下，NLP展现出巨大的应用潜力。

NLP主要利用循环神经网络（RNN）。循环神经网络（Recurrent Neural Network, RNN）是一种特殊类型的人工神经网络，专门用于处理序列数据。与传统的前馈神经网络不同，RNN具备“记忆”能力，能够通过其内部状态（隐藏层）对输入序列中的先前信息进行存储和利用。这一特性使得RNN在处理时间序列、自然语言处理和语音识别等任务时表现出色。

RNN的基本构造单元是循环结构，这允许信息在网络中循环流动，从而使得当前的输出不仅依赖于当前的输入，还依赖于之前的输入。这种特性使得RNN能够捕捉数据中的时间依赖关系。例如，在语言模型中，RNN可以根据上下文信息生成下一个单词，从而实现流畅的文本生成。

循环神经网络及其变种在处理序列数据方面具有广泛的应用，包括机器翻译、语音识别和情感分析等领域。随着深度学习技术的发展，RNN在各种实际应用中展现出越来越重要的角色，推动了自然语言处理和其他领域的进步。RNN结构如下所示：



基于这样的背景，我提出了基于文本表头和文本内容的交叉验证的数据分类算法，其基本思想如下。对于文本表头，依然使用与传统方法相似的字符串模糊匹配的算法进行匹配，对于直接能够匹配到相似字段的表头内容，则不进行内容的检测匹配，直接将其划分。对于无法进行匹配的表头内容，我们进入下一阶段的内容匹配。对于表内容进行基于内容的文本分类。表内容是一系列的字符串，基于这些字符串，我们需要利用NLP和神经网络相结合的处理方法，在本项目中，我所用算法如下：

1. 利用自然语言处理的一些Python包对表内容的字符串进行处理，将其转化为一系列的向量，我选用gensim与nltk。

2. 选取神经网络模型，进行分类。由于不同的表内容，其字符串的长度是不一样的，因此为了把所有的字符输入到我们的模型中。我们必须要选用一种能够接受不同长度输入的模型，我选用的是循环神经网络模型

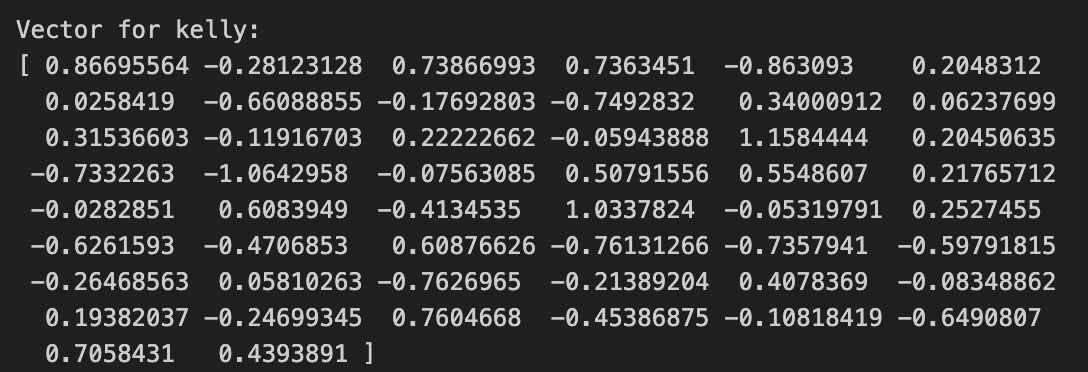
3. 最后，训练神经网络，完成数据分类

1. 算法
2. 字符串预处理：  
   在本次试验中，由于数据集缺乏，我首先用faker库自行生成了个人信息数据集，部分结果如下：

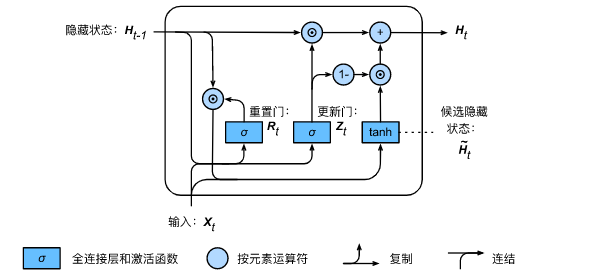


我们希望根据表内容直接对其进行分类。注意到表内容都是一系列的字符串，因此为了利用RNN，首先把字符串用分隔符分裂开，转化为一个单词数组，例如把出生日期1986-04-17划分为：[‘1986’,’04’,’17’]。接下来把所有生成的单词构成一个大的语料库，利用gensim中的word2vec方法，构建单词到向量的一一映射，并将预处理后的原始数据划分为训练集、验证集和测试集。

一个例子：



1. 模型  
    RNN有多种cell，在程序中我实现了基本的RNN cell、GRU cell和LSTM cell。后两种性能较好。根据实验最好的是GRU cell。其结构如下：



神经网络的训练和推断与一般无异，都是在torch框架下进行的。在程序中很容易看到。

1. 程序结构与使用方法

目前暂时的1.0版本，其结构如下：

1)数据生成:data\_gen.ipynb，生成的数据在raw\_training\_data.csv中

2)数据预处理，将每个单词转化为向量: pre\_process.ipynb

3)数据预处理演示，可视化词向量，显示相似的词有更接近的词向量:pre\_process\_demo.ipynb

4)用GRU单元的RNN模型进行训练(普通RNN单元和LSTM单元也可以稍加修改选择): training.ipynb, 训练后的最佳模型存储在 ./trained\_model/checkpoint\_run.pth中

5)模型加载和示例显示: result.ipynb，所显示的示例存储在./example\_results.csv文件中

6)其他:model.py和RNNcell.py是定义我自己的RNN网络的重要文件，分别是RNN模型结构与不同的RNN cell

7)结果示例:在example\_results.csv中给出了一些预测示例

8)训练好的模型在 ./trained\_model中

9)用于训练的数据在./data中

使用方法：

1）First run data\_gen.ipynb to generate raw\_training\_data.csv

2）then run pre\_process.ipynb to get ./data

3）then run training.ipynb to train model

4）finally run result.ipynb for some results

环境配置：

python==3.8.0

Faker==27.0.0

gensim==4.3.3

numpy==1.24.4

scipy==1.10.1

sklean==1.3.2

torch==2.2.2

1. 实验结果与参数选取  
   一些我进行的优化：
2. GRU cell的选取以及结构参数的设置
3. Adam优化器的选取与学习率的设置
4. 数据鲁棒性的提升
5. Batch size的设置
6. 验证集的参与，避免过拟合现象的发生

参数选取：

1. 生成数据集的长度：10000\*22
2. 训练集、验证集、测试集的划分：8：1：1
3. 编码的词汇向量的长度：50
4. 训练batch size：32
5. 验证batch size：320
6. 训练轮数：10
7. 学习率：0.001
8. GRU隐藏层大小：64

代码仓库：

https://github.com/liziming18/data-classification

结果示例：

