深圳大学考试答题纸

(以论文、报告等形式考核专用)

二〇 二四 ～二〇 二五 学年度第 一 学期

课程编号 15190001 课序号 05 课程名称 人工智能与机器学习 主讲教师 xx 评分

学 号 姓名 专业年级

教师评语：

题目：基于 Transformer 模型和决策树的冷链库产品价格预测系统

基于 Transformer 模型和决策树的冷链库产品价格预测系统

【摘要】本研究旨在探讨如何利用深度学习模型 Transformer+决策树对冷链库产品价格进行准确预测。

关键词：冷链库产品；价格预测；Transformer；决策树；深度学习

【正文】

第一章 Transfomer 及决策树简介及相关工作

1.1 简介

本系统主要的工作流程是 PCA 进行特征选择之后，用 Transformer 对对应的特征进行时序预测，再把预测的特征输入到决策树模型中实现对未来价格的预测。

在本研究中，Transformer+决策树模型被结合使用，以提高冷链库产品价格预测的准确性。Transformer 模型是一种基于自注意力机制的神经网络，主要用于处理序列到序列的任务，如机器翻译。它由编码器和解码器组成，能够很好地捕捉输入数据的全局信息。

具体来说，Transformer 模型通过多头自注意力机制建模输入序列中不同位置之间的关系，而其编码器部分则将输入序列转化为特征表示。这些特征表示随后通过决策树模型进行价格预测。决策树是一种有监督的学习算法，通过对数据特征进行分类来构建数学模型，进行数据筛选和决策。这个过程通过递归划分数据集，实现对数据的分类和预测。

研究表明，Transformer 在复杂特征处理和缺失值处理上表现优异，而决策树模型则提升了整个模型的灵活性和预测准确度。这种结合优化了特征的提取及后续的价格预测过程，为冷链行业提供了更高效的预测解决方案。

1.2 相关工作

(1)特征选择(数据探索与可视化)

在数据的初步探索中，我们进行了对数据的统计分析，包括数据的均值、方差、最大值和最小值等指标的计算。我们还使用可视化工具对数据进行了可视化分析，散点图，以帮助我们更好地理解数据的特征和趋势。通过数据的探索和可视化分析，我们可以更好地把握数据的规律性和潜在的关联关系，为模型的训练和预测提供有价值的参考。

下面是我使用 PCA 在 matlab 上对数据进行特征分析的结果。

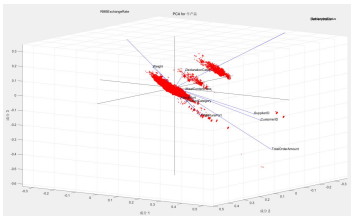

Figure 1 牛产品

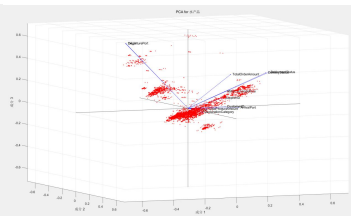

Figure 2 水产品

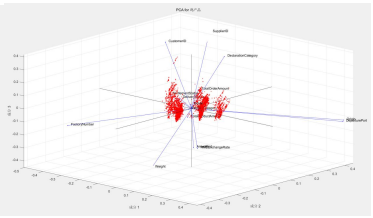

Figure 3 鸡产品

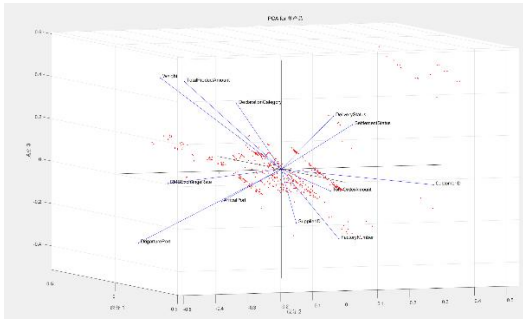

Figure 4 羊产品

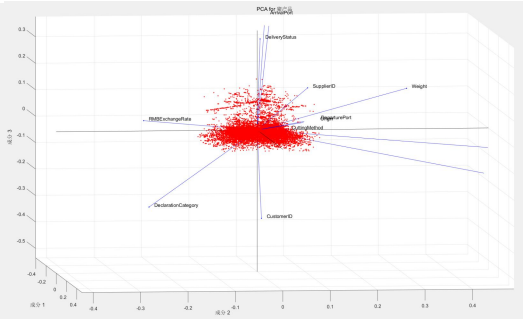

Figure 5 猪产品

我们可以很清楚的从三维散点图中看到清晰的特征分层现象，这个对我们后面的特征筛选有很大的帮助。

(2)数据的预处理

Transformer 模型数据预处理过程是为了确保数据的准确性、一致性和可用性，以便在 Transformer 模型训练之前进行进一步的数据分析和处理。例如，数据筛选和转换可以确保只使用了特定商品大类的数据，并将单价转换为统一的货币单位。标签编码可以将文本数据转换为数值数据，方便模型处理。最后，计算平均值可以提供更加稳定和可解释的特征用于模型训练。具体操作步骤如下：

首先，读取 xlsx 格式的数据集的数据，并将其存储在名为 df 的 DataFrame 中，分步筛选出"商品大类"的行数据(牛产品,水产品,鸡产品,羊产品,猪产品)，保留下来，然后对"单价(原币)"列的数值乘以"人民币汇率"列的数值，得到新的"单价(原币)"列的数值。这一步是为了将单价转换为统一的货币单位。

将特征选择中分层明显的特征维度的数据类型转换为字符串类型以为了后续的标签编码准备，接着使用 LabelEncoder 函数对不同数据类型的特征进行了标签编码。标签编码是将文本标签转换为数值编码的过程，方便后续的数据分析和建模。

从 df 中选择了对应的特征列和"采购时间"、"单价(原币)"，并重新赋值给 df。这一步是为了保留我们需要的列数据，去除了其他不需要的列。将"采购时间"列的数据类型转换为日期时间格式。这样可以方便后续的时间序列分析和处理。使用 groupby 函数对数据进行分组，按照"采购时间"列进行分组，并计算其他列的平均值。计算得到的平均值存储在名为 result 的 DataFrame 中。

最后，使用 result 的 to_excel 方法，将计算得到的平均值存储到名为"out.xlsx"的 Excel 表格中。

由于数据的尺度不同，不同的特征可能具有不同的取值范围，例如某个特征取值范围在 0-1000，另一个特征取值范围在 0-1。如果不进行归一化，可能会导致某些特征对

模型的影响更大，而忽略了其他特征的重要性。

因此存入表格之后，我们对数据进行归一化将数据缩放到相同的范围，以便更好地适应模型的训练和推理。

通过使用了 MinMaxScaler 函数将数据的范围限定在[-1, 1]之间。

这样做不仅可以提高模型的收敛速度和稳定性，还有助于加快模型训练的速度并提高模型的泛化能力。

(3) 特征预测

使用处理好的数据对特征基于 Transformer 进行时序预测。

(4) 价格预测

将预测的特征输入决策树模型中开始价格预测。

……

第二章 模型原理及方法分析

2.1 Transformer+决策树模型的原理

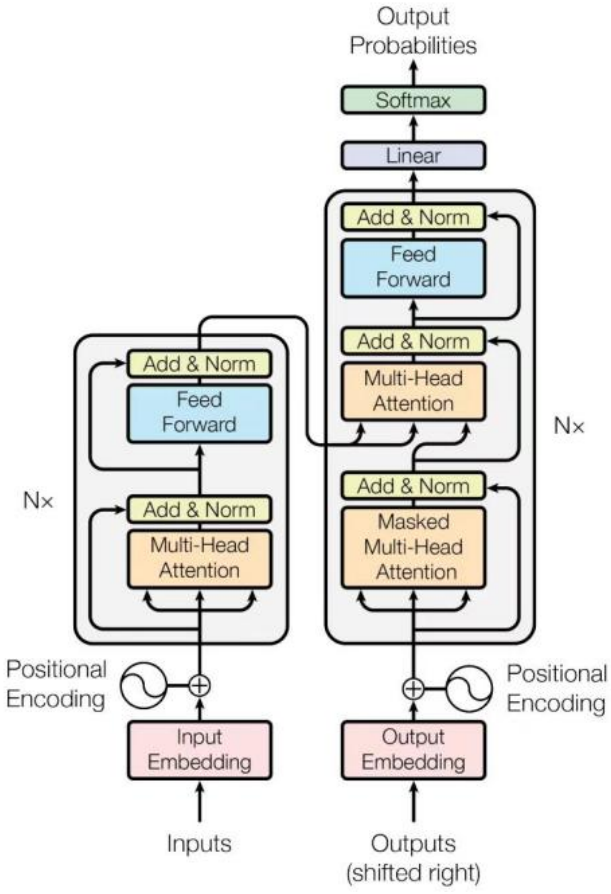

Figure 6 Transformer 原理图

（1）Transformer 模型是一种基于注意力机制的神经网络模型，用于处理序列到序列的任务，例如机器翻译。它由编码器和解码器组成，每个部分都由多个层级组成。以下是 Transformer 模型的原理：

位置编码：Transformer 模型不能利用单词的顺序信息，因此需要在输入中添加位

置编码。位置编码是一个向量，它与输入嵌入相加以表示单词在序列中的位置。

自注意力机制：Transformer 模型使用自注意力机制来计算输入序列中每个单词的表示。自注意力机制允许模型在计算每个单词的表示时，同时考虑输入序列中的所有单词。

编码器：编码器由多个层级组成，每个层级包含一个自注意力子层和一个前馈神经网络子层。自注意力子层计算输入序列中每个单词的表示，前馈神经网络子层对这些表示进行处理。

解码器：解码器也由多个层级组成，每个层级包含一个自注意力子层、一个编码器-解码器注意力子层和一个前馈神经网络子层。自注意力子层计算目标序列中每个单词的表示，编码器-解码器注意力子层允许解码器在生成每个单词时，参考编码器中的输入序列。

注意力机制：注意力机制允许模型在计算每个单词的表示时，参考输入序列中的其他单词。自注意力机制和编码器-解码器注意力子层都使用注意力机制。

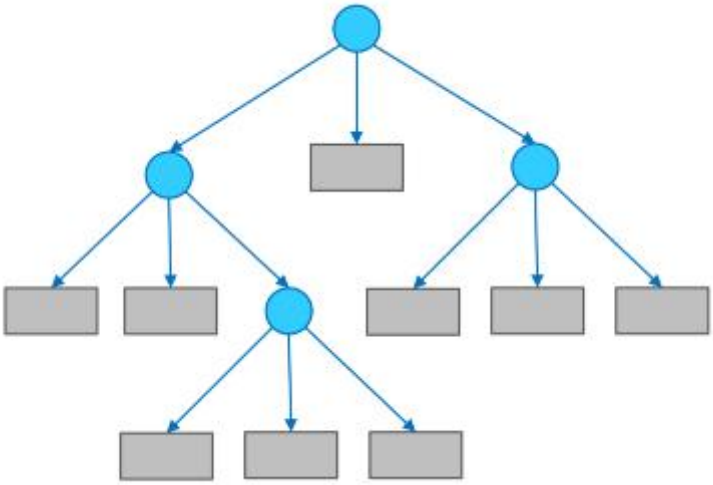

Figure 8 决策树原理图

（2）决策树模型是一种有监督学习算法，它通过对数据的特征进行分类，构建数学模型，从而实现数据的筛选和决策的目标。决策树的生成过程是一个递归的过程，它从根节点开始，根据数据的特征将数据集分成不同的子集，然后对每个子集重复这个过程，直到所有的数据都被分类到叶子节点上。在决策树的生成过程中，需要选择一个合适的特征作为划分标准，使得划分后的子集尽可能的纯，即同一子集内的数据尽可能的相似，不同子集之间的数据尽可能的不同。这个过程可以通过计算信息增益或者基尼指数来实现。在决策树的剪枝过程中，需要考虑全局最优，通过对决策树进行剪枝，可以避免过拟合的问题。

总的来说

Transformer 模型基于自注意力机制和多层感知机（MLP）构建。它使用多头自注意力机制来对输入序列中的不同位置之间的关系进行建模，并通过多层感知机来进行非线性变换和特征提取。

在特征预测阶段，Transformer 模型将输入数据作为序列输入，并通过编码器部分将其转化为特征表示。编码器根据输入序列中的信息学习出每个位置的特征表示。
预测的特征表示通过决策树模型进行价格预测。决策树模型是一种基于树结构的机器学习模型，它根据特征的取值进行划分，并根据划分后的数据进行决策。

综上所述，使用 Transformer 模型先预测特征再用决策树模型预测价格的好处在于提高了模型的灵活性和预测准确度。Transformer 模型可以更好地处理复杂特征和缺失值，并提供更准确的特征表示，而决策树模型则可以利用这些特征进行价格预测。

2.2 对于 Transformer 模型应用

首先，我们定义了一个 PositionalEncoding 类，用于对输入数据进行位置编码。在类的初始化方法中，我们创建了一个位置编码张量 pe，其形状为(max_len, d_model)。我们使用 torch.arange 函数创建了一个表示位置的张量 position，然后使用 torch.exp 函数创建了一个表示位置编码的除法因子 div_term。接下来，我们通过对 pe 张量的奇数列和偶数列分别应用正弦和余弦函数，将位置编码的奇偶性信息编码到张量中。最后，我们通过对 pe 张量进行维度变换和缩放操作，将其形状调整为(1, max_len, d_model)，并将其注册为模型的缓冲区。

接着，我们定义了一个 TransAm 类，用于实现 Transformer 模型。在类的初始化方法中，我们设置了模型的类型为"Transformer"，并初始化了源序列掩码 self.src_mask。我们使用 PositionalEncoding 类创建了一个位置编码器 self.pos_encoder，并使用指定的特征大小 feature_size 和层数 num_layers 创建了一个 Transformer 编码器层 self.encoder_layer。然后，我们使用 self.encoder_layer 和指定的层数 num_layers 创建了一个 Transformer 编码器 self.transformer_encoder。最后，我们使用一个全连接层 self.decoder 将 Transformer 编码器的输出映射到目标变量的维度。我们还定义了一个 init_weights 方法来初始化全连接层的参数。

在模型的前向传播方法 forward 中，我们首先判断源序列掩码 self.src_mask 是否需要更新，并根据源序列的长度生成一个方形的上三角掩码。然后，我们将输入序列 src 通过位置编码器 self.pos_encoder 进行位置编码。接着，我们将位置编码后的序列输入到 Transformer 编码器 self.transformer_encoder 中，并根据源序列掩码 self.src_mask 进行编码。最后，我们将 Transformer 编码器的输出通过全连接层 self.decoder 进行映射，并返回输出结果。

接下来，我们定义了一个 create_inout_sequences 函数，用于生成输入和输出序列对。该函数接受输入数据 input_data 和时间窗口大小 tw 作为输入。在函数内部，我们通过循环遍历输入数据生成输入序列和对应的输出序列，其中输入序列 train_seq 由当前时间窗口的前 tw-output_window 个元素和输出窗口大小 output_window 个零值组成，输出序列 train_label 由当前时间窗口的前 tw 个元素组成。最后，我们将生成的输入和输出序列转换为 PyTorch 的 FloatTensor 类型，并返回结果。

然后我们定义了一个 get_data 函数，用于读取 Excel 文件并获取训练数据和测试数据。在函数内部，我们使用 pd.read_excel 函数读取 Excel 文件，并获取时间序列和要预测的变量。然后，我们使用 scaler.fit_transform 函数对变量进行归一化处理。接着，我们根据整个数据集的长度计算出训练数据的样本数量，并将训练数据和测试数据分割开。最后，我们调用 create_inout_sequences 函数生成训练数据和测试数据的输入和输出序列，并返回结果。

接着进行数据准备，首先根据给定的数据源，通过 get_batch 函数获取一个批次的数据。数据源是一个序列数据，通过指定的索引 i 和批次大小 batch_size，确定每个批次中的序列长度 seq_len，并从数据源中提取相应长度的数据。数据的输入窗口大小由 input_size 指定。

之后进行模型训练，将模型切换到训练模式：model.train()，以便模型能够进行训练。通过 for 循环，获取每个批次的数据。将数据输入模型进行前向传播：将获取的数据作为模型的输入，通过调用模型的 forward 方法得到输出结果 output。根据指定的损失函数 loss_fn，计算输出结果 output 与真实值 target 之间的损失 loss。通过调用 optimizer 的 backward 和 step 方法，对损失进行反向传播，并更新模型参数。打印当前批次的损失值 loss 和训练速度 speed。

我们还对损失曲线进行绘制，将模型切换到评估模式：model.eval()，以便模型能够进行评估。通过 for 循环，获取每个样本数据。将获取的数据作为模型的输入，通过调用模型的 forward 方法得到输出结果 output。根据指定的损失函数 loss_fn，计算输出结果 output 与真实值 target 之间的损失 loss。将输出结果 output 和真实值 target 保存下来，以便后续绘制图形。使用 pyplot 库绘制三个图形：预测结果图、真实值图

和预测结果与真实值之间的误差图。将绘制完成的图形保存为图片文件。

对未来预测方面我们将模型切换到评估模式：model.eval()，以便模型能够进行预测。根据给定的步数 steps，依次预测未来的值，获取一个输入窗口大小的数据 input。将输出窗口部分置为0。将 input 输入模型进行前向传播得到输出结果 output。将 output 添加到数据中作为下一个输入数据。重复上述步骤，直到预测完指定步数的未来值。使用 pyplot 库绘制两个图形：预测未来值图和输入值图。
将绘制完成的图形保存为图片文件。

最后我们进行模型评估，将模型切换到评估模式：model.eval()，以便模型能够进行评估。通过 for 循环，获取每个批次的数据。将获取的数据作为模型的输入，通过调用模型的 forward 方法得到输出结果 output。根据指定的损失函数 loss_fn，计算输出结果 output 与真实值 target 之间的损失 loss。返回平均损失值作为模型的评估指标。

transformer 模型的应用过程包括数据准备、模型训练、损失曲线绘制、未来预测和模型评估等步骤。其中，模型训练过程包括前向传播、损失计算、反向传播和梯度更新等操作。损失曲线绘制、未来预测和模型评估利用模型的输出结果进行可视化分析和性能评估。我们通过 transformer 模型预测出了未来的特征，之后我们选择了决策树来使用预测出来的特征进行了价格预测。

决策树的模型训练，使用 DecisionTreeRegressor 类初始化一个决策树回归模型 regressor,并设置 random_state 为 0 以保持结果的可重复性。使用 X_train 和 Y_train 作为训练数据,通过调用 fit()方法对模型进行训练,其中 X_train 是特征数据,Y_train 是对应的目标值。

决策树的模型预测，使用训练好的模型对新的特征数据 X_test 进行预测，通过调用 predict()方法，将预测结果赋值给 y_pred。其中 X_test 是特征数据，y_pred 是预测的目标值。

然后我们的可视化结果，设置可视化参数，如字体大小、字体类型等。使用 matplotlib 库绘制折线图，横轴表示样本的索引，纵轴分别表示真实值和预测值。添加图例，设置标题，并显示绘制的图形。

对于模型应用，读取待预测的特征数据 x_pred，通常为 Excel 或 CSV 文件。提取所需的特征列，如'等级'、'到达港'和'供应商 ID'，并将其转换为 numpy 数组格式。使用训练好的模型对 x_pred 进行预测，通过调用 predict()方法，将预测结果赋值给 y_pred1。

接着结果保存，创建一个字典 data，将预测结果 y_pred1 存储在键名'单价'下。使用 pd.DataFrame 将字典 data 转换为 DataFrame 格式，并赋值给 df_output。使用 to_excel()方法将 df_output 保存为 Excel 文件，其中指定保存到的列为第一列，设置 index=False 以避免保存索引，指定 engine='openpyxl'来使用 openpyxl 引擎。

.....

第三章 实验论证

3.1 价格预测结果

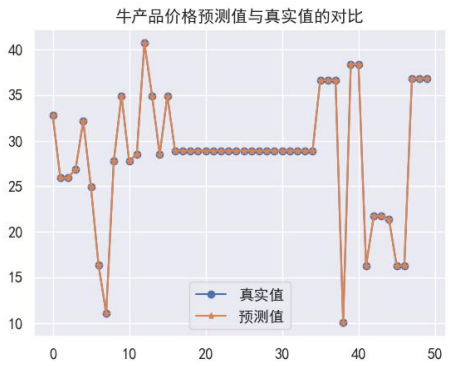

Figure 8 牛产品价格预测值与真实值的对比

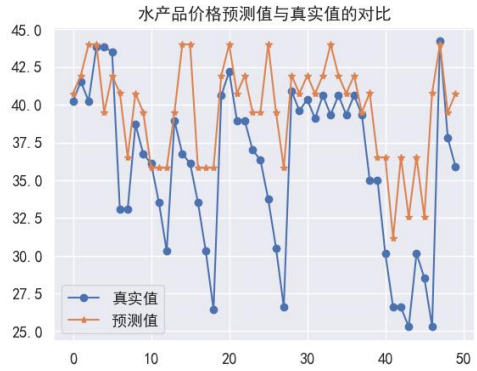

Figure 9 牛产品价格预测值与真实值的对比

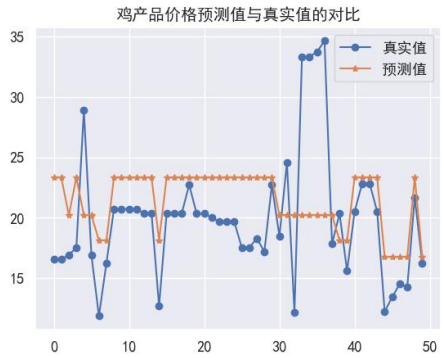

Figure 10 鸡产品价格预测值与真实值的对比

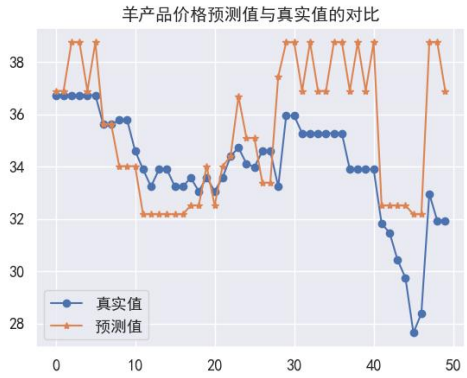

Figure 11 羊产品价格预测值与真实值的对比

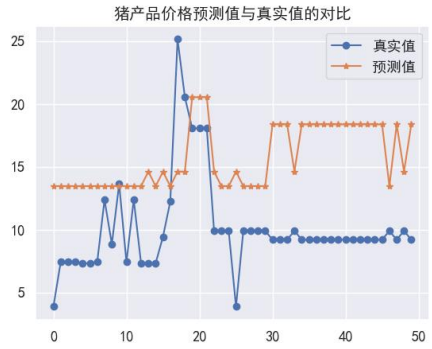

Figure 12 猪产品价格预测值与真实值的对比

由上面几幅图可以看出，模型在牛产品方面的价格预测是比较准确的，而在其他商品大类上得到的结果稍微差一点。

……

【结论】（谈谈自己的看法和理解）

模型适用性及局限性讨论

适用性：

高维特征处理: PCA (主成分分析) 非常适合处理高维特征数据。在许多金融时间序列预测问题中，特征数量庞大（例如，各种经济指标、市场情绪指标等），PCA 可以有效地将高维数据降维到低维空间，同时保留大部分重要信息，从而减少计算量并避免维度灾难。

捕捉长期依赖: Transformer 模型擅长捕捉时间序列数据中的长期依赖关系。这对于价格预测尤其重要，因为价格往往受到过去较长时间内各种因素的影响。与循环神经网络 (RNN) 相比，Transformer 可以并行处理数据，训练速度更快，且能处理更长的序列长度。

非线性关系建模: 决策树模型擅长处理非线性关系。将 Transformer 的预测结果输入决策树，可以更好地捕捉价格与预测特征之间的复杂非线性关系，从而提高预测精度。

局限性：

PCA 的信息损失: PCA 降维过程中不可避免地会损失一些信息。如果丢弃的信息与价格预测密切相关，则会影响预测精度。PCA 的效果很大程度上取决于数据的分布和特征之间的相关性。

Transformer 的计算成本: Transformer 模型，尤其是在处理长序列时，计算成本较高，需要强大的计算资源。这可能会限制其在资源受限的环境中的应用。

Transformer 的可解释性: Transformer 模型是一个“黑盒”模型，其预测结果的可解释性较差。难以理解模型是如何做出预测的，这会增加模型风险，尤其是在需要对预测结果进行深入分析和解释的场景下。

决策树的过拟合风险: 决策树模型容易过拟合，尤其是在训练数据量不足或特征数量较多时。过拟合会导致模型在训练集上表现良好，但在测试集上表现较差，降低了模型的泛化能力。

【参考文献】（逐个举出自己用到的参考资料）

1. [2202.07125v2] Transformers in Time Series: A Survey (arxiv.org)

…….

【附录】

1．字数统计

2．补充说明 （可选）