Can Foundation Models Wrangle Your Data 复现研究报告

刘畅

2024.1.10

摘要

当前大语言模型能够在文本处理和自然语言对话任务中达到惊艳的成果，因为其蕴含的参数和模型的推理能力，大语言模型能够胜任通用的领域。但对于数据处理和分析领域，尚且存在一些难以解决的问题，通常需要较为专业的数据分析师来对海量数据进行分析。然而，大语言模型因为具有通用性，因此在一定程度上可以胜任例如实体解析、实体匹配、数据填充、错误检测等任务。

本文从实验的角度，尝试复现利用大语言模型进行数据治理的流程。但使用现有的参数较少的，可以运行在本地的通用大语言模型进行该任务并不能做到非常好的数据治理效果，因此本文尝试使用经过低秩矩阵微调的模型进行上述数据治理任务。相比于通用语言模型，微调后的模型在处理数据这一特定领域上，能够达到更好的治理效果。

关键词：大语言模型；数据治理；数据挖掘；模型微调

1 引言

当前大语言模型在自然语言处理和问答场景下，能够发挥相当大的作用，人们已经见证了大模型在通用领域的强大之处。而在数据处理和分析领域，尚且存在一些难以解决的问题，对于大部分非专业的数据分析师来说，想要对海量的数据进行分析，这是相当困难的。然而，利用通用的大模型，也可以做到对数据的处理和清洗。需要将数据按照一定的格式编码，并给出一定的背景和环境，然后向大模型提出问题，大模型将会按照给定的提示词，做出相应的回答。

在这种情况下，用户可以指定大模型输出处理完成的数据，或者是用来处理和清洗数据的一段 SQL 代码 [2]，有指向性地解决问题。该论文从一系列问题的建立入手，从实体匹配、实体检测、数据填充等方面 [5]，执行数据清洗和处理的任务，并尝试了不同模型，分别对于数据处理的成果评分。

在另一方面，随着 ChatGPT 商业化的趋势不断增强，现有诸多服务商有能力提供集中的大语言模型访问服务。与此同时，该论文的数据治理方法依赖于在线大语言模型服务，例如 GPT-3 等，而对于需要保密的隐私数据，使用在线的大语言模型服务来进行这部分的数据治理可能会有泄露数据的风险。本文在尝试复现该论文对数据集进行治理的同时，也尝试将

语言模型本地化，通过本地调用接口的形式，能使得数据治理通过本地模型来运行，在大大降低了需要进行网络数据交换造成的数据隐私泄露的风险。

2 相关工作

2.1 传统的数据治理方法

传统的数据治理方法通常是一些基于规则的，核心思想是统计学中的抽样，是通过分析样本数据来得到结论，重点在于抽取样本的规则合理性。传统数据分析是对已知可理解数据的分析，而且数据大多是与业务直接相关的，是对已知数据的一个精细化挖掘的过程。

而大数据分析是全量数据分析，很多数据之间并没有正式的可理解的关系。对于各种类型，不同业务的数据，若采用传统的数据治理方法，通常需要动用相当一部分的人工，这会造成非常高昂的成本。

2.2 基于机器学习的数据治理方法

数据治理方法中，通常包含实体匹配、实体检测、数据填充、错误检测、数据集成等层面的操作。数据清洗通常包含处理缺失数据、删除重复数据、数据的一致性等问题，使得数据集中的每一条记录都能有高质量保证。数据集成通常包括将数据从多个数据源导入并对齐到一个共同的格式，用于后续数据的处理或模型的训练。

基于机器学习的数据治理的方式，通常是针对特定领域的数据，通过机器学习中的多层感知网络或图神经网络等方式，将已经标注好的数据训练成一个领域应用特定的模型，从而解决该特定的问题。然而，这些方法论，我们都并不认为其是严格的数据治理任务，而是一个监督学习的机器学习任务。本文所需要达成的目标是一种在比较通用的领域中，能够基本正确地执行实体解析、实体匹配、数据填充、错误检测等方面的任务。

现有工作中，例如 ValueNet [1]，LGESQL [2] 都达到了将自然语言文本转换为 SQL 查询的目标，且这些方法在挑战性较高的 Spider [6] 数据集上均达到了较好的效果。其中，ValueNet 使用数据记录、数据库表 schema、用户提示词这 3 种数据类型，通过神经网络为该任务生成一种基于语法分析树的中间表示，从而得到目标任务的 SQL 语句。LGESQL 利用图神经网络获取自然语言中的多跳语义，通过注意力机制，学习给定任务中的隐含信息，从而构建目标 SQL 语句。

3 本文方法

3.1 本文方法概述

本文主要面向实体匹配 (EM)、错误检测 (ED)、数据填充 (DI) 这几方面的工作。数据集 D 具有 n 个入口，即 n 条记录，每条记录表示为 m 个属性映射到值的二元对。对于数据集的每个入口 $e_i \in D$，都有 m 个属性，表示为 $e_{i,j} = \{\texttt{attr}_j, \texttt{val}_j\}$。

实体匹配任务将不同数据集中的实体进行匹配，该问题通常按照分类问题来处理。形式化表示为：对于给定的两个结构化数据集 (D, D')，其中每个实体对 $e, e' \in D \times D'$，目标是预测这两个实体是否相同。

错误检测任务通常在给定数据记录时，检测某个属性的值是否存在错误，通常按分类问题来处理。传统方法中，错误检测通常是基于规则的算法、或者是基于统计分析等方式解决。该问题的形式化表达为：给定一个入口 e，检测参数 $\texttt{attr}_j$ 的值 $\texttt{val}_j$ 是否存在错误。

数据填充任务是修复脏数据的关键步骤，已有方法中通常采用基于统计和聚类方式、基于生成式模型、基于机器学习和深度学习方式来完成。该问题的形式化表达为：给定一个入口 e，以及其中缺失的数据 $\{\texttt{attr}_j, \texttt{NULL}\}$，生成数据来填充 NULL 值。

3.2 数据序列化模块

大语言模型的输入一般为纯文本，而对于数据处理任务，结构化表格数据必须转换为序列数据，才能够将该序列作为大语言模型的输入。因此，本文按照以下的方式，将每条需要传输给大语言模型的数据按照格式序列化为文本数据。

$$\texttt{serialize}(e) := \texttt{attr}_1 : \texttt{val}_1, \ldots, \texttt{attr}_m : \texttt{val}_m$$

对于数据填充任务中空数据，本文将其序列化为空字符串。基于给定的任务和数据集，数据的序列化只会运行在少部分的数据记录上，而并不会将完整的数据表全部转换后传输给语言模型做长文本的处理。在该步骤中，模型将会对序列化后的提示词产生敏感，并生成对应任务的回复。

3.3 任务构建模块

在数据记录序列化的基础上，还需要人工添加该任务相关的提示词，同时还需要指定语言模型生成的数据格式，以便后续自动化对该任务的处理。一般地，本文将序列化后的数据记录按照一定的模板填充到指定位置，成为提示词，整体作为模型的输入。

实体匹配任务给定两个实体 (e, e')，任务的模板为

$$\texttt{Product A is serialize}(e)\texttt{. Product B is serialize}(e').$$
$$\texttt{Are Product A and Product B the same?}$$

数据填充任务给定一个入口，一个需要填充的数据，模板为

$$\texttt{attr}_1 : \texttt{val}_1, \ldots, \texttt{attr}_j :?$$

错误检测任务给定一个入口，一个待检测的字段，模板为

$$\texttt{Is there an error in attr}_j : \texttt{val}_j\texttt{?}$$

构建模板后，直接将问题输入给模型，并不一定能让模型输出的结果满足实验目的。如果提供的提示词过少，模型将无法正确地回答问题，相反，会因为其“幻觉”而生成许多我们并不希望出现的结果。因此，还需要为模型提供一些问答的样例，以增强语言模型对问题的理解能力。

4 复现细节

4.1 与已有开源代码对比

原文采用的方法是不对大语言模型进行任何修改，只通过提示词，来对已经序列化的数据来进行分析和处理。本篇工作在提示词上进行了部分优化，构建格式更加确定的序列化方式。原文采用的是由 OpenAI 提供的大语言模型服务进行请求式数据处理，这对于一些需要保密的隐私数据来说，可能会有泄露的风险，因此，本文尝试使用本地的体积较小的模型来执行数据治理的任务。

在此基础上，本文尝试使用本地大语言模型进行数据治理，但参数量较低的模型未能很好地按照指定的提示词来生成期望的结果。因此，本文尝试采用基于 OpenOrca-Platypus2-13B，在数据治理任务上微调的模型 Jellyfish-13B，仿照原文逻辑结构，重新构建提供给模型的提示词输入结构，使得模型能够输出符合实验目标的结果。

4.2 实验环境搭建

实验采用 Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz 共 32 核处理器，376GB 内存。GPU 使用 NVIDIA A30 图形运算单元，对大语言模型运行加速。编程语言环境选择 Python 3.11.5，使用 PyTorch 和 transformers 第三方库进行模型的加载。

本地大语言模型使用多种方式部署

- 使用 Ollama 框架，将模型部署为本地 RESTful 服务，使用 HTTP 请求可以获取指定模型的响应
- 使用 PyTorch 加载 huggingface 已有的大语言模型，初始化参数、权重后，使用模型进行问讯

4.3 实验运行流程与方法

本文使用原文提供的数据集。在实体匹配任务中，每个数据集包含 table A 和 table B 表，分别用于两类不同数据记录的实体匹配。实验可通过指定参数，来控制当前运行何种数据治理任务，通过 data access module，将训练集、验证集、测试集读取到数据对象中。

通过参数设置，可以为当前数据处理任务添加 k 个样例，用于 few-shot 少样本学习。指定模型参数，限制 token 长度，可以在输出层面上做一些限制，从而在一定程度上控制当前任务的输出结果。

在提示词模板构建方面，采用原文构建的 few-shot 提示词，组成完整的提示词模板，从而输入语言模型，得到输出结果。该部分能够从实验验证的角度，证明通过大语言模型的能力可以在一定程度上做到数据处理的任务。

(Instruction) Now I am executing a entity matching task. Please follow the Q&A below and answer the question. Are Product A and Product B the same? Yes or No? Please answer 'Yes' or 'No' only, not need to give explanations.
(Few shot) Product A is title: automata theory for xml researchers. authors: frank neven. venue: sigmod record. year: 2002. Product B is title: automata theory for xml researchers. authors: frank neven. venue: very large databases year: 2002. Are Product A and Product B the same? No
(Question) Product A is title: an annotated bibliography on real-time database systems. authors: özgür ulusoy. venue: sigmod record. year: 1995. Product B is title: secure buffering in firm real-time database systems. authors: binto george , jayant r. haritsa. venue: the vldb journal – the international journal on very large data bases. year: 2000. Are Product A and Product B the same?

(Response) No

4.4 创新点

使用本地大语言模型并进行自动化处理，并不一定能够得到较为精确的结果。尤其是在提供的样本较少，或者上下文环境尚未构建时，大语言模型通常会先进行冗长的分析，从而给出一个结论。当该部分任务涉及到大语言模型未能掌握的领域特定的知识时，程序将会生成一段分析，不给出明确的结果。原文中，作者的对生成的结果进行人工筛选，从而去除了自动化处理中未能精确处理的输出。在这一层面上，通用的大语言模型虽然有能力做数据治理的任务，包括一些特定领域的实体解析、数据填充等问题，但又受限于模型本身的特性，会基于给定的文本数据，来生成较长的文本结果。

本文针对该分析，考虑使用 LoRA [4] 对模型进行微调，使大模型能够适用于实体解析、数据记录错误检测、缺失数据填充等任务。在微调任务中，使用数据集中部分标注好的数据进行处理，微调数据集样本如表 1所示。

实验中，采用 [Instructions, Few shots, Question, Answer] 的格式来构成数据集，从而对语言模型进行微调。该过程包括调整以更好地理解和响应特定领域相关的问题。数据结构允许模型学习上下文相关信息，使其能够提供更准确和深刻的反应。在数据集的准备过程中，考虑按照上述的 [Instructions, Few shots, Question, Answer] 的格式组织为 (问题, 答案) 对，通过 PEFT 库进行微调，增强其生成逻辑和信息响应的能力。模型微调的有效性是基于其产生上下文恰当的响应能力来评估的，从而呈现出该模型在数据治理方面的价值。

Task	Dataset	Instances	Positives
ED	Adult	550	35
	Hospital	1250	40
DI	Buy	586	N/A
	Restaurant	600	N/A
SM	Synthea	2500	15
EM	Amazon-Google	359	31
	Beer	359	54
	DBLP-ACM	359	71
	DBLP-GoogleScholar	359	69
	Fodors-Zagats	359	41
	iTunes-Amazon	359	92

表 1. 微调各数据集样本数量，Instances 为样本数量，Positives 为正样本数量

5 实验结果分析

使用本地 Llama2 模型进行一定任务的数据治理结果如表 2所示。在不进行模型微调的情况下，实体匹配任务和数据表 schema 匹配任务上，模型能够拥有较高的召回率，但其他指标普遍较低。这表明该模型在大多数情况下，对正样本的识别能力较强，但很难较为准确地识别负样本。

dataset	task	prec	rec	f1	acc
Amazon-Google	entity matching	0.035533	0.777778	0.055274	0.040000
Beer	entity matching	0.144444	**0.928571**	0.250000	0.142857
Buy	data imputation	0.000000	0.000000	0.000000	0.076923
DBLP-ACM	entity matching	0.075000	**1.000000**	0.139535	0.075000
DBLP-GoogleScholar	entity matching	0.055276	**0.916667**	0.101340	0.055000
Fodors-Zagats	entity matching	0.116402	**1.000000**	0.208531	0.116402
Restaurant	data imputation	0.000000	0.000000	0.000000	0.290698
Synthea	schema matching	0.036269	**0.875000**	0.063472	0.065000
Walmart-Amazon	entity matching	0.025000	**1.000000**	0.048780	0.025000

表 2. 直接使用本地 llama2 模型进行任务处理，各项任务结果评估

使用本地 OPT-6.7B 模型进行上述任务的数据治理结果如表 3所示。由于 Open Pre-trained 语言模型是实验性的项目，较少参数的模型并未达到较为理想的效果，在一些实体匹配的任务中，只要单纯的预测为正确，则基本上能够达到实验数据中占正样本的比例。

dataset	task	prec	rec	f1	acc
Amazon-Google	entity matching	0.066667	0.111111	0.014815	**0.890000**
Beer	entity matching	0.125000	**0.846154**	0.211538	0.122222
Buy	data imputation	0.000000	0.000000	0.000000	**0.968750**
DBLP-ACM	entity matching	0.093750	0.200000	0.037500	**0.795000**
DBLP-GoogleScholar	entity matching	0.000000	0.000000	0.000000	**0.940000**
Fodors-Zagats	entity matching	0.318182	**0.954545**	0.477273	0.755319
Hospital	error detection	0.000000	0.000000	0.000000	**0.860000**
iTunes-Amazon	entity matching	0.000000	0.000000	0.000000	0.750000
Restaurant	data imputation	0.000000	0.000000	0.000000	**0.825000**
Walmart-Amazon	entity matching	0.026738	**1.000000**	0.052083	0.090000

表 3. 直接使用 OPT-6.7B 模型进行任务处理，各项结果评估

使用微调后的模型进行数据治理的结果如表 4所示。在模型使用低秩矩阵微调过后，相比于前面未执行微调的模型来说，在准确度、召回率上都有了非常明显的提升。

dataset	task	prec	rec	f1	acc
Amazon-Google	entity matching	0.333333	0.777778	0.466667	**0.920000**
Beer	entity matching	0.378378	**1.000000**	0.549020	0.747253
Buy	data imputation	0.000000	0.000000	0.000000	**0.907692**
DBLP-ACM	entity matching	**1.000000**	**1.000000**	**1.000000**	**1.000000**
DBLP-GoogleScholar	entity matching	0.611111	**0.916667**	0.733333	**0.960000**
Fodors-Zagats	entity matching	**0.880000**	**1.000000**	**0.936170**	**0.984127**
Hospital	error detection	0.027397	**0.800000**	0.043836	0.285000
iTunes-Amazon	entity matching	0.642857	**1.000000**	0.782609	**0.862385**
Restaurant	data imputation	0.000000	0.000000	0.000000	**0.755814**
Walmart-Amazon	entity matching	0.500000	**1.000000**	0.666667	**0.975000**

表 4. 使用微调后的模型进行数据治理，各项结果评估

6 总结与展望

大语言模型在自然语言问答的场景中能够拥有比较惊人的能力，在通用领域中，大语言模型能够在大部分情况下，做出严谨和严格的分析，从而得出其推论。大语言模型的推理能力有助于在数据治理任务中，实施实体匹配、实体解析、数据填充、错误检测等任务，相比于现有的基于传统数据治理方法和基于机器学习的治理方式上，能够提供更好的推理原因。

然而，受限于大语言模型本身的性质，其擅长的工作为文本推理、对给定的文本进行合理的解释。而本文的工作为，向大语言模型提供一系列文本描述和任务，使得模型能够生成一个单独的输出，用于表明该任务的最终结果。但事实上，大语言模型擅长的是对给定的上

下文进行分析，在逐步推导中得出结果 [7]，因此其可以作为一个辅助的推理器，在推理完成的最后得到最终结果。

另外，对于参数量较少的模型来说，其对于上下文的记忆通常会不够准确。在不进行微调的大语言模型对数据治理任务的实验中，针对某些数据记录，模型将不会按照提示词中的说明首先给出答案。因此，在这些条件下，针对数据治理方面的模型微调是比较有必要的。

未来的工作中，需要针对数据集和数据任务，对该任务进行有效的数据集摘要，作为大语言模型的一部分输入。同时，需要根据总结的已有的提示词，构建出对大语言模型较好的提示词，即提示词重写。另外，根据任务的复杂程度，可以将数据治理任务做一些划分，简单任务可以通过生成能够完整处理该任务的代码来完成，而困难任务可以考虑使用大语言模型的推理能力来完成 [3]。

参考文献

[1] Ursin Brunner and Kurt Stockinger. ValueNet: A Natural Language-to-SQL System that Learns from Database Information. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2177–2182, Chania, Greece, April 2021. IEEE.

[2] Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations, June 2021. arXiv:2106.01093 [cs].

[3] Zui Chen, Lei Cao, Sam Madden, Tim Kraska, Zeyuan Shang, Ju Fan, Nan Tang, Zihui Gu, Chunwei Liu, and Michael Cafarella. SEED: Domain-Specific Data Curation With Large Language Models, October 2023.

[4] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021. arXiv:2106.09685 [cs].

[5] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. Can Foundation Models Wrangle Your Data? *Proceedings of the VLDB Endowment*, 16(4):738–746, December 2022.

[6] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, 2019.

[7] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. Jellyfish: A Large Language Model for Data Preprocessing.