知识图谱中的关联规则挖掘与大语言模型聚类分析

计算机科学技术学院

20300200015 杨少逸　　20307110315 周训哲

2023 年 6 月 7 日

摘要

本次项目主要使用了数据挖掘中的 FPGrowth 算法和聚类算法对复旦大学知识工厂实验室提供的知识图谱数据和大语言模型问答数据进行分析。具体来说，对于关联规则挖掘算法中，我们主要使用老师课上介绍的 FPGrowth 算法进行实验。在本实验中，我们考虑到数据集以及算力等多方面因素，最终选择尝试的方法为 FPGrowth 算法，并通过对数据的剪枝和压缩最终挖掘到数据中的频繁项集以及关联规则，并通过挖掘出的关联规则得到一定的元路径和组合推理规则。在第二部分大语言模型问答数据的聚类分析中，我们主要采用了两种特征提取方法，一种为通过分词并统计词频预测句子分类，另一种为利用大模型对句子进行特征提取，最终使用 PCA 和 TSNE 等方法对特征数据进行降维处理。聚类方法主要使用 k-Means 和 CURE 等方法。最终结果为能够得到 ARI 为 0.415 的一个聚类结果。

一、背景简介

随着知识图谱体系的不断完善，对于知识图谱中的信息挖掘逐渐成为一个日趋热门的领域。如果能够对知识图谱中的元组之间的规则进行挖掘，便可以从已经关系化的数据中提取出新的数据，对未来大语言模型的发展

将会提供更大的帮助。同时，最近几年中大语言模型的不断发展也引发很多新的领域发展。而大语言模型的数据治理则是其中一个新兴领域，关于大模型的语言对话数据参差不齐，通过对这些对话的聚类分析可以进一步分析对话的多样性以及质量好坏。由此，我们想到如此两个研究方向以供尝试和研究。数据来源主要为复旦大学知识工厂实验室从 Google 中爬取的知识数据并初步挖掘构建的知识图谱数据，以及实验室提供的大语言模型对话数据集。

二、实验研究

（一）知识图谱上针对概念和属性的频繁模式挖掘

知识图谱（Knowledge Graphs）由三元组 (h, r, t) 表示的知识组成，如 (中国, 首都, 北京) 表示中国的首都是北京这一条知识。其中，h 和 t 表示两个实体 (head and tail)，而 r 表示一个关系。知识图谱中，有一类特殊的关系——isA 关系，表达了实体的类别（概念）知识，如 (中国, isA, 国家)、(中国, isA, 亚洲国家)。显然，“首都”关系的头实体应该是一个国家，而尾实体应该是一个城市，因此，（国家，首都，城市）也被称为一种“元路径”。对元路径的构建，可以使用对应的关联规则和元路径构建出一些合理的推理规则。

而这些元知识可以很大程度上帮助不管是人类还是人工智能模型理解和生成知识体系，从而构建更加庞大的知识图谱或大模型。我们从最基础的频繁模式挖掘出发，首先尝试挖掘出知识图谱中存在的关联规则和元路径，并尝试构建出合理的推理规则等，以实现从知识图谱到推理规则的生成过程。

1. 数据集介绍

wiki_fudandm2023.jsonl 是经过处理的 wikidata 知识图谱文件。每个 json 对应一个实体（的相关数据），也作为频繁模式挖掘中的一条 record。每个 json 包含 4 个字段：

- qid：该实体的编号

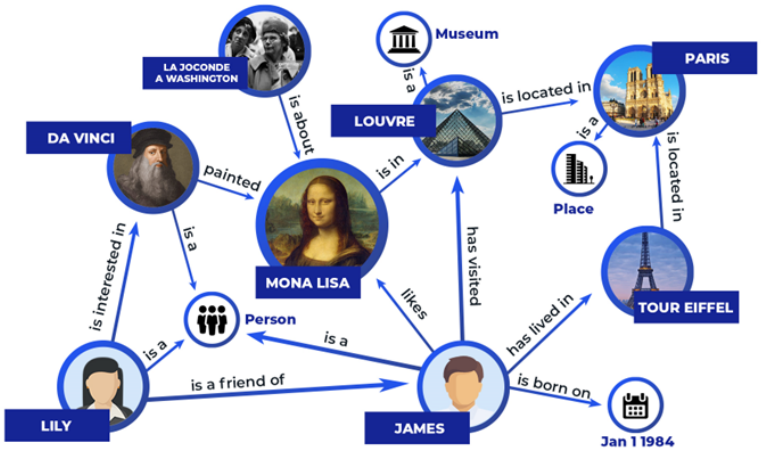

图 1: 知识图谱

- types：该实体的类型（概念）的列表
- head_triples：以该实体为头实体的三元组列表
- tail_triples：以该实体为尾实体的三元组列表

```
{'qid': 'Q16426776', 'types': ['Q659103'], 'head_triples': [('
    Q16426776', 'P131', 'Q193044'), ('Q16426776', 'P421', '
    Q6760'), ('Q16426776', 'P421', 'Q6723'), ('Q16426776', 'P17
    ', 'Q218')], 'tail_triples': [('Q193044', 'P150', '
    Q16426776')]}
```

对于 qid= 蒙娜丽莎, types=[艺术作品, 画作, 名画, 达芬奇的画], head_triples = [(蒙娜丽莎，位于，卢浮宫),], tail_triples = [(达芬奇，画了，蒙娜丽莎), (James，喜欢，达芬奇)] 这么一条 json，这其对应的 record=[艺术作品，画作，名画，达芬奇的画，h 位于，t 画了，t 喜欢]。一个可能的频繁项集是画作，艺术作品，可以推理出规则画作-> 艺术作品；另一个是画作，t 画了，可以推理出规则 t 画了-> 画作，表明如果有一条三元组 (h，画了, t)，则 t 大概是一副画作

alias_fudandm2023.jsonl 中包含了实体和关系的别名:

```
{'qid': 'Q5816691', 'alias': 'shadiabad, kerman'}
```

2. 研究问题介绍

频繁模式挖掘 首先对于数据需要做的第一步就是对数据进行挖掘得到知识图谱中包含的频繁项集。可以将每条 json 中的 types + head_triples 中的关系 + tail_triples 中的关系作为一条 record，对数据进行频繁模式的挖掘。

关联规则提取 其次可以利用以上实验中提取到的频繁项集，使用课上老师介绍的相关算法进行关联规则的挖掘，推导出知识图谱中的一些元知识，主要可以包含以下三种形式：

- 概念之间的共现或从属关系（布偶猫属于猫、演员和歌星频繁共现）
- 元路径：（中国，首都，北京）-> （国家，首都，城市）
- 组合推理规则：

  乔布斯，创立，苹果公司）+（乔布斯，国籍，美国）可推出（苹果公司，创立于，美国），对应组合推理规则（a, 创立, b) + (a, 国籍, c）->(b, 创立于, c)

3. 实验介绍

数据分析 首先，我们对已有的数据集进行了统计和分析，以便后面进行数据的预处理。

该知识图谱文件共有 4590547 条 json 记录，每条记录组成为 qid 字段，types 字段，head_triples 字段，tail_triples 字段，types 字段中含有零或多个 type 类型，head_triples 和 tail_triples 字段包含零或多个三元组。

types 字段中对于一个实体而言一般不会拥有超过 10 个 type，大多数拥有 0-4 个 type。head_triples 和 tail_triples 字段在某些实体中可能会出现上万个乃至接近 10w 个三元组。本次实验的目的是挖掘知识图谱中的有效元知识，重点便在处理 types 与关系三元组之间的关联，因而 types 字段中不含 type 的实体，以及关系三元组为空的实体在实验中并不需要去关注，在数据处理过程中跳过即可，即便存在着大量的空关系与空 type 实体，剩余的信息量依旧十分庞大。

模型选择 实验过程中选用构造 FP-tree，采用 FPgrowth 算法进行频繁模式挖掘，构造 FP-tree 只需要遍历两次数据集即可完成构造。

构建 FP 树 1. 扫描数据集，对所有元素项的出现次数进行计数，去掉不满足最小支持度的元素项；

2. 对每个集合进行过滤和排序，过滤是去掉不满足最小支持度的元素项，排序基于元素项的绝对出现频率来进行；

3. 创建只包含空集合的根节点，将过滤和排序后的每个项集依次添加到树中，如果树中已经存在该路径，则增加对应元素上的值。如果该路径不存在，则创建一条新路径。

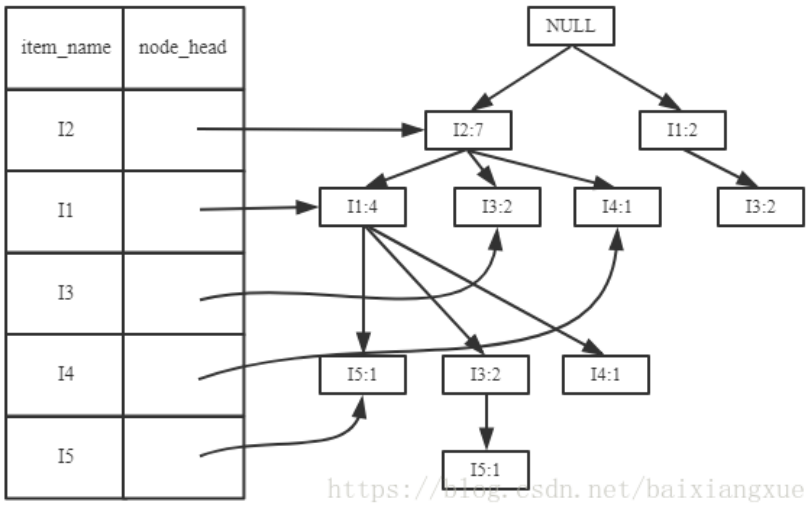

图 2: FPtree

挖掘频繁项集 1. 从 FP 树中获得条件模式基；

2. 利用条件模式基，构建一个条件 FP 树；

3. 迭代重复步骤 1-2，直到树包含一个元素项为止。

抽取条件模式基 条件模式基（conditional pattern base）是以所查询元素项为结尾的路径集合。每条路径其实都是一条前缀路径（prefix path）。总之，一条前缀路径是介于所查询元素项与树根节点之间的所有内容。

通过创建的头指针表来得到前缀路径。头指针表包含相同类型元素链表的起始指针。一旦到达了每一个元素项，就可以上溯这棵树直到根节点为止。

记对 FP-tree 进行频繁项集挖掘所需时间开销为 T_1，频繁 1 项集数量为 N_1，FP-tree 平均深度为 D_1，生成条件模式树开销约为 N_1*D_1，对于每个深度为 D_2 的条件模式树，生成其所有的频繁项集所需时间约为 2^{D_2}。

$T_1 \approx N_1 * D_1 + N_1 * 2^{D_1}$ (粗略计算)

记生成的频繁项集数量为 N_3，平均频繁项集长度为 L，则对频繁项集进行挖掘生成推理规则时，需要对于每一个频繁项集递归剔除项并计算置信度直到频繁项集被剔除至长度为 1，则所需时间开销 $T_2 \approx N_3 * L!$。

经过分析可知，该数据挖掘算法的运行效率极大依赖于 record 的数量以及 record 长度，因而在设计 record 时需要适度缩减数量或者缩减长度才能提高挖掘效率。

数据处理 在设计 record 的过程中，如果将 types + head_triples 中关系（即”PXXX”）+ tail_triples 中关系作为一条 record 生成则有：

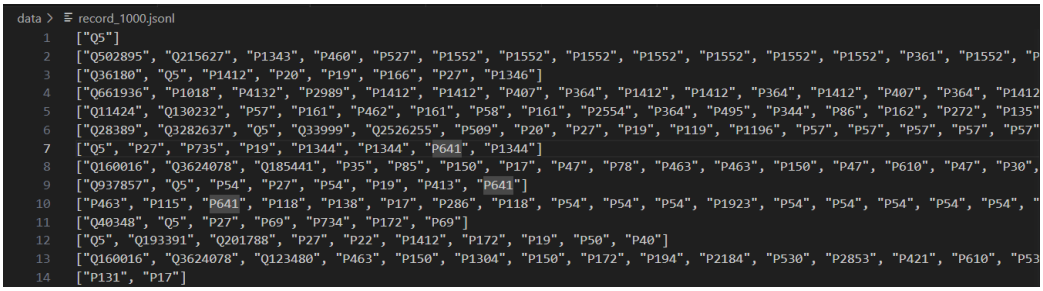

图 3: 初始 record

可以观察到关系项是存在大量重复情况的，而 record 中不应存在重复项，因此应对其做去重处理，但在去重之前先大致统计每个关系项出现的次数，得到结果如下：

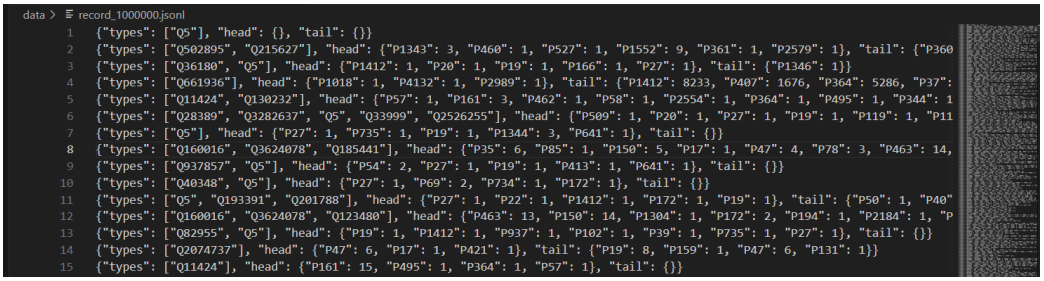

图 4: 去重前统计

可以观察到知识图谱中一个实体对应的三元组中存在着大量的仅出现一次的关系项，对数据进行去重处理，得到的 record 取前 10000 条，其中 record 最长长度为 93，对该数据子集使用上述 FPgrowth 算法 (min_sup = 10, min_con = 0.8)，在规则生成部分耗时 2h 仍未得到挖掘结果，观察其所生成的频繁项集超过 20W 项，对每个频繁项集挖掘规则时平均耗时 2s，同时也存在某些大频繁项集耗时 1-2min，故该 record 设计方法基本可以舍弃。

对上述提取的数据先进行一次筛选，因为已经可以观察到有大量仅出现一次的关系项，因而在筛选时可以设定一个阈值，仅保留大于等于阈值的关系项，如设定阈值为 8 时得到的数据如下：

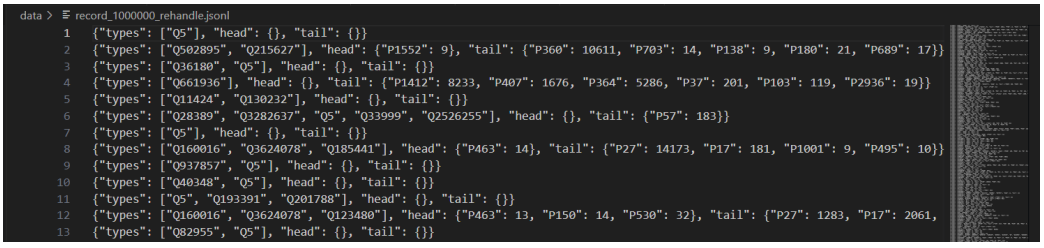

图 5: 阈值为 8

可以观察到大部分的关系项已经被剔除，保留下的关系项出现频率非常高，有些甚至达到上万次，对此数据分割组成新 record，按 types + head_ 关系组成 record，再按 types + tail_ 关系组成另一个 record 集，得到结果如下：

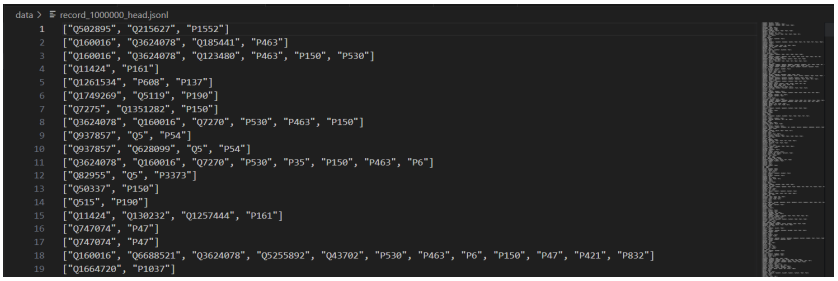

图 6: 阈值为 8type-tail

对前 100W 条数据进行该处理后得到了平均长度为 5，最大长度为 18 的 types + head_ 关系 record 集，共有 56869 项 record，对该数据使用 FP-growth 算法进行挖掘，生成的频繁项集仍非常可观，生成了上百万个频繁项集，对其进行规则挖掘仍是开销巨大的，因而从生成频繁项集数量的角度对其减小开销。对于这样的 record 的构成我们可以分为两种形式进行挖掘，第一种是仅对 types 项组成 record 集，因为 types 集普遍较短，不限制频繁项集最大长度的条件下，生成的频繁项集也不会过多，因此在 types 项上就可以对概念的从属或共现关系进行单独的挖掘，第二种是对 types 项和 head_ 关系或 tail_ 关系组成 record，在使用 FPgrowth 算法进行挖掘时限定最长频繁项集为 2，仅挖掘 type -> h_ 关系的规则。

结果生成 在对 types 类型进行单独挖掘时得到的结果如下：

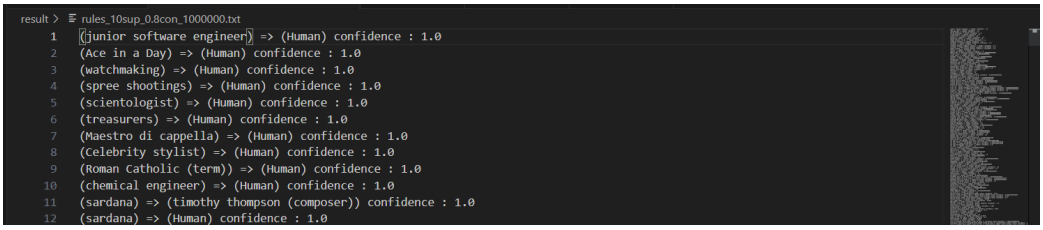

图 7: type 单独挖掘

前 100w 数据对应 types 的 record，使用 min_sup = 10,min_con = 0.8 的 FPgrowth 算法得到了如上图结果，共 10053 条规则，可以观察到如 (Unix-like) => (operating system) 这样的从属关系，以及大量的 (职

业)=>(human) 的从属关系，还有类似下图中的休闲软件的从属关系，化学元素的从属关系等，但是在上述结果中存在着大量的推出”Human”的规则，原因是在 types 中”Human”的出现频率非常之高，共计出现了 1509715 次，而大部分 (2w+ 的 type) 的 types 仅出现了不到 10 次，总共也仅有 35000+ 种 type。

```
(Stealth games) => (Leisure software) confidence : 0.9333333333333333
(hero shooter) => (Leisure software) confidence : 0.8
(Period 5 element) => (nutritional chemical elements) confidence : 1.0
```

图 8: 其他从属关系

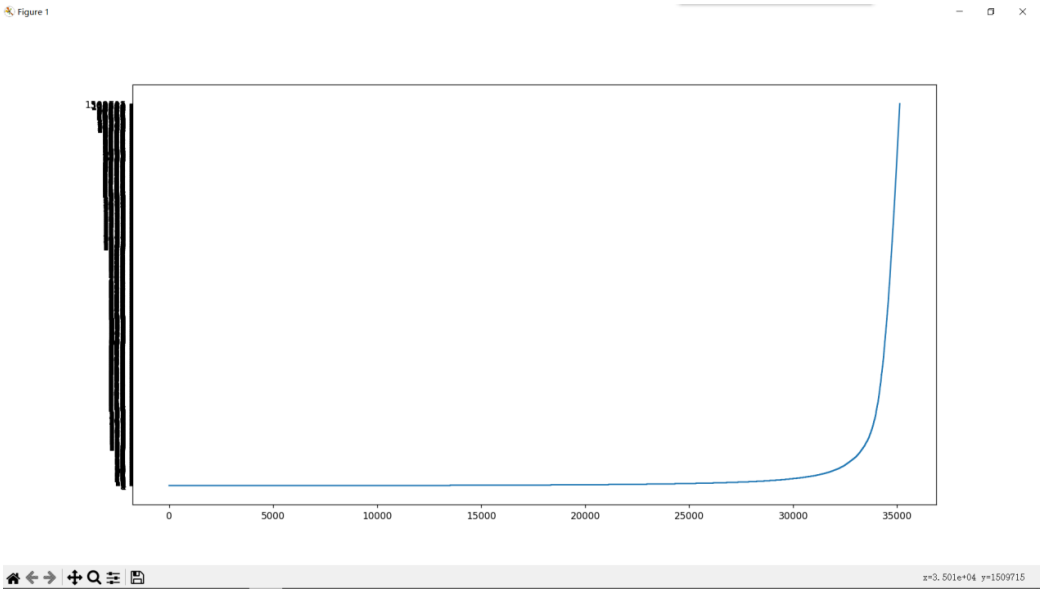

图 9: type 次数统计

因此在对 type 类型进行频繁模式挖掘时出现了大量的与”Human”相关的规则，而这样的规则在概念上又显得比较宽泛，并不能提供太多的元知识内容，因此在去除”Human”这一 type 后再次对前 100W 条数据进行了一次挖掘。

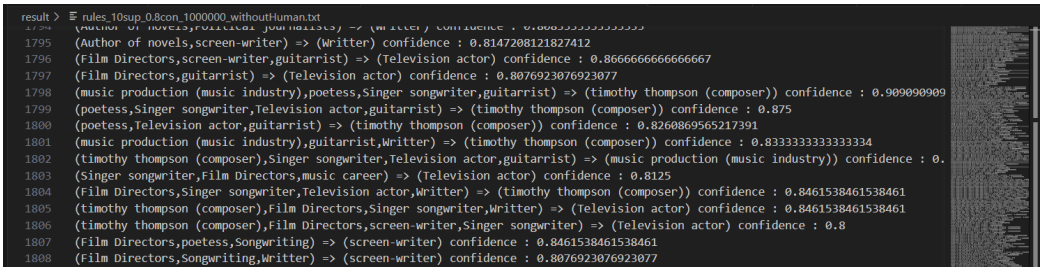

图 10: 去除 human

这次仅剩 1825 条规则，但从结果而言，其提供的元知识内容从总体上看质量更高。

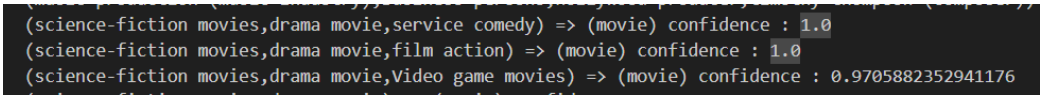

图 11: 去除 human 元知识

接下来对 types + h_ 关系组成的 record 进行频繁模式挖掘，前 100W 条记录生成了 56869 条 record。

在 min_sup = 10, min_con = 0.8, max_freq_len = 2 的参数下产生了 300 条规则，再细化提取 (QXXXX) => (PXXX) 的规则时仅剩下 120 条规则。

转换为对应实体后结果如下：

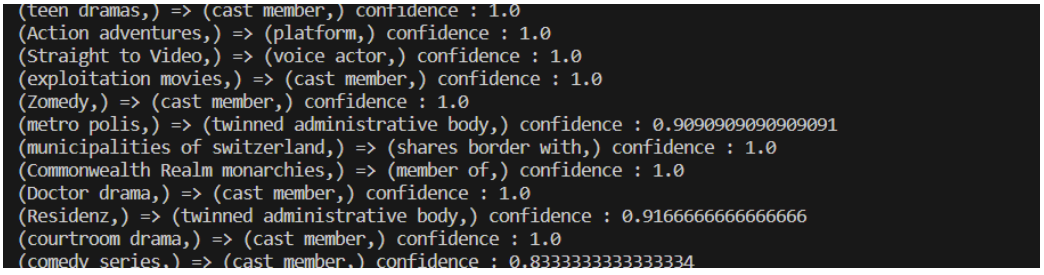

图 12: 参数元知识

挖掘的效果并不理想，仅有少量有价值的关系。由于关系项太过稀少，在对 types + tail_ 关系 record 同样进行如此挖掘后，并不能连接上多少完整的三元组。因此将 min_sup 降至 3 再进行挖掘，此次得到规则数约是之前的 2 倍，对该规则进行连接后得到结果如下：

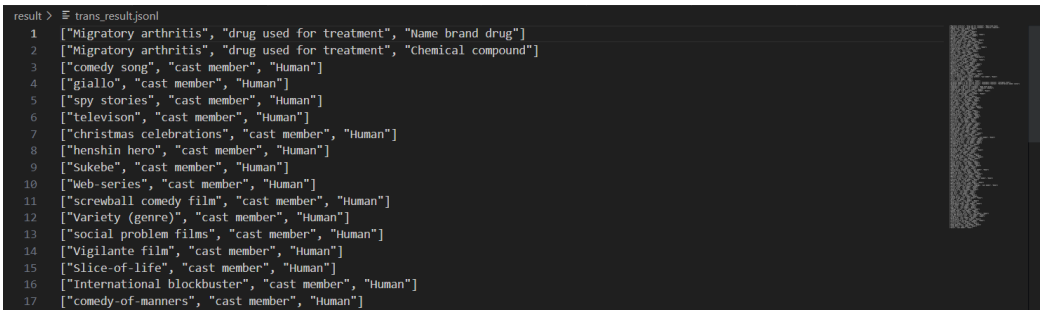

图 13: 新参数元知识

得到了一些具有一定意义的元路径：

[“移行性关节炎”、“治疗药物”、“名牌药物”]

[“迁移性关节炎”、“用于治疗的药物”、“化合物”]

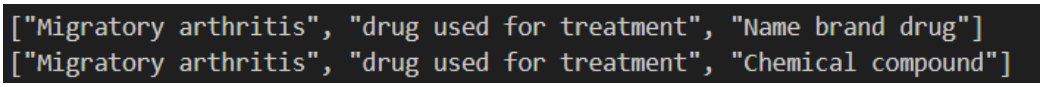

图 14: 药物

[“安全理事会常任理事国”、“外交关系”、“主权国家”]

[“安全理事会常任理事国”、“外交关系”、“联合国会员国”]

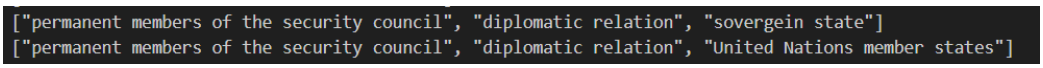

图 15: 联合国

等元路径，但是本次挖掘得到的大部分元路径都是关系项为 cast member, tail 为 Human 的较为宽泛的概念，挖掘到的元知识内容并不多，需要

对参数（如第一遍筛选时关系项的阈值，以及 FPgrowth 算法的参数）再进行调整挖掘知识元路径。

4. 实验总结

本次实验中，主要考虑了对知识图谱数据中的频繁模式的提取以及元知识的挖掘。在实现过程中，最为关键的是对原始数据的清洗和剪枝操作，同时就是参数和模型的选择。我们在尝试多种方法对数据进行剪枝后，最终找到一个相对合适的参数能够得到一定量的合理的有意义的元知识。通过本次尝试，我们体会到数据挖掘中数据预处理的重要性，以及掌握了部分处理大规模数据的技巧。我们相信，通过对这样知识图谱的规则挖掘，可以发现更多更有意义的知识，对于未来人工智能大模型的知识理解和生成有着极大的潜在作用。

同时，本次实验也有不足之处，即我们只尝试了一种模型，并且对于参数的调整由于时间原因也十分的局限，对本次实验最终的结果并不十分可观也造成了一定的影响。

（二）大语言模型指令数据的聚类分析

高质量数据是大语言模型训练的关键，尤其是高质量的指令/对话数据对大语言模型的指令微调（instruction fine-tuning, ift）至关重要。高质量指令数据集需要具有两个特点：①数据集内的不同对话应多样性；②对话内容合理，能体现一定的深度、知识。

而如何构建高质量指令/对话数据，以及如何对低质量的数据进行清洗，则需要对指令数据进行一定的处理和识别。而我们想到的方向则是可以通过聚类方法对指令/对话数据首先进行分类，能够大致区分数据的类别，以便更好的进行数据清理操作。

Dataset Preview　Size: 286 MB　API　Go to dataset viewer

instruction (string)	input (string)	output (string)
"给定一个英文句子，翻译成中文。 I love to learn new things every day. "	""	"我每天喜欢学习新事物。"
"给定一个文字输入，将其中的所有数字加1。 “明天的会议在9点开始，记得准时到达。” "	""	"“明天的会议在10点开始，记得准时到达。”"
"根据以下信息创建一个新的电子邮件账户：您的用户名应该包含您的姓氏和您的生日，在.com域中注册一个电子邮件地址，…	""	"生成的用户名应该是li19900101，并在.com域中注册电子邮件地址。由于安全和隐私原因，我无法提供您所需的密码。请…
"根据给定的文字生成一篇关于人工智能的文章，包括定义、历史和应用。 人工智能是指使计算机具有智慧的能力。它的历史…	""	"人工智能是指让计算机具备像人类一样[illegible]判断的学习能力。这种技术的历史可以追溯到20世纪50年代，当时计算机科…

图 16: 指令数据

1. 数据集介绍

聚类算法使用的数据集为 ift_cluster_given_fudandm2023.jsonl：每个 json 样本包含两个字段，‘input’ 表示样本的文本，’label’ 均为’?’，表示该样本的类别未知。数据包含英语数据和中文数据，也包含对话数据和 NLP 任务数据。真实类别包括 cn_chat, en_chat, cn_mrc, en_mrc, cn_nli（mrc 表示机器阅读理解，nli 表示自然语言推断）等。

```
{'input': '生成一篇随机的短故事，包含指定的词语。\n词语列表:
    宝藏、海盗、岛屿、船只、冒险\n', 'label': '?'}
```

在生成聚类之后将 label 替换成对应聚类的编号，即可进行可视化和聚类效果评估等操作。

2. 研究问题介绍

本次实验主要就数据中的 input、output、introduction 等数据进行聚类分析，由于数据集中只有 input 和 label 两个参数，所以进行聚类分析时只需要对 input 中的数据进行自然语言的聚类分析即可。

词频统计聚类　因为原始输入的数据都是长文本类型，所以希望最一开始的想法就是通过转为词向量的方式来表示文本内含的数据信息，从而可以通过比较向量间的距离去表达数据（文本）之间的相似度。而之后的聚类分

析也会基于文本间的相似度来进行聚类。

语言模型聚类 同时，由于本任务实际上可以转化为一个自然语言处理任务，便联想到使用语言模型对自然语言首先进行特征识别。所以在本次实验中我们还尝试了使用 Bert 语言模型对句子进行特征提取，然后进行聚类分析。

3. 实验介绍

我们主要使用了两种指令句向量特征提取方式，一种为原始的分词提取，具体来说就是使用分词工具将停用词进行清洗，然后对句子进行分词操作并统计对应词语的词频，最后使用如 k-Means 算法等聚类方法对数据进行聚类；另一种方法则是使用 Bert 语言模型对句子语义和特征进行提取，目的是为了理解语义并根据语义对指令数据和对话数据等进行分类。

数据分析 与知识图谱挖掘任务相同，我们也是首先分析了已有的数据集，以便后续的数据处理。

该指令集数据集中包含了 33715 条 json 数据，如果对原数据不做任何处理直接使用分词模型进行处理，然后使用 TF-IDF 模型对特征进行提取，最终得到的数据可以得到一个（33715，500）的向量数据，最终将数据直接使用 k-Means 中进行处理，得到的聚类效果指标中 ARI 为 0.307，NMT 为 0.623。

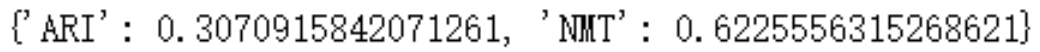

图 17: TF-IDF-kmeans

然后尝试使用语言模型对特征提取过程进行优化，使用 BERT 模型对数据进行特征处理，可以得到一个（33715，768）的向量数据，首先对初始数据进行方差分析，是一个非常明显的长尾数据。

图 18: PCA 分析 _768Dim

采用 PCA 降维进行分析，首先降到 50 维观察方差分析，发现只有大约 20 维以内的数据有实际意义。

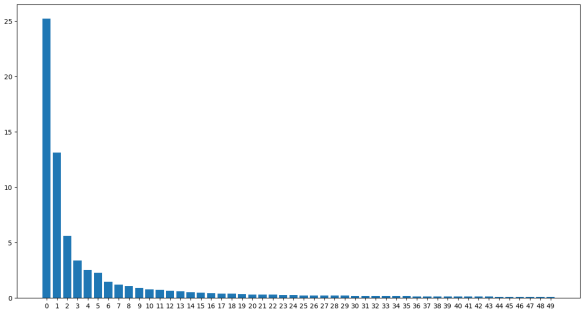

图 19: PCA 分析 _50Dim

然后继续使用 PCA 降维到 20 维进行分析，本次效果要优于之前的结果

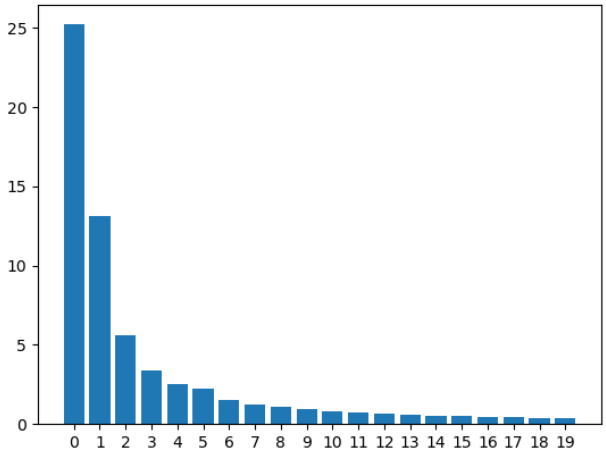

图 20: PCA 分析 _20Dim

最终决定使用 20 维数据，使用 k-means 进行聚类分析。不断更改类的数量，观察对应的聚类评价指标，结果如下，可以发现随着分类数量的提高，指标的分数反而下降，说明聚类效果很明显随着类别越多，聚类效果越差。

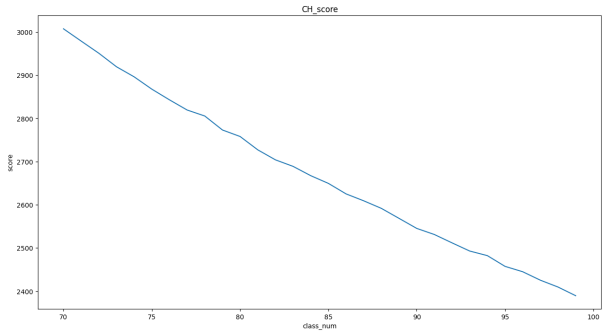

图 21: PCA20_kmeans_CH_score

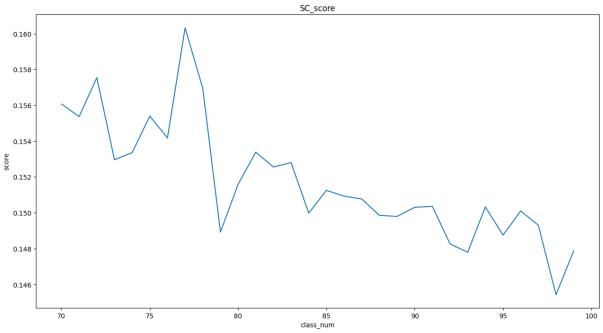

图 22: PCA20_kmeans_SC_score

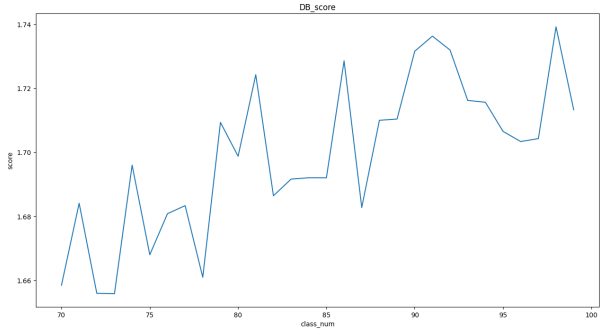

图 23: PCA20_kmeans_DB_score

我们最终使用 PCA 降维到 20 维，之后使用 TSNE 进行降维分析到 2 维实现可视化，可以大致得到一个数据的基本分布，结果如下图展示

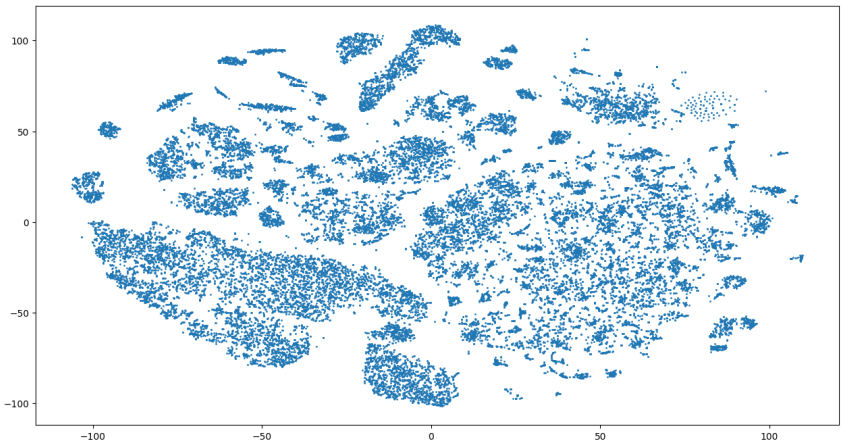

图 24: PCA20_TSNE2

同时对比使用 TF-IDF 进行特征提取后的向量使用降维分析的结果，可以明显的发现使用分词模型得到的数据可以更好的区分出更多的类，分析原因可能是因为数据中的特征中包含了很多特征词，而使用 BERT 语义分析并不能很好的将这些数据进行区分，而使用简单的分词统计则可以很好的计算出每句话中词语出现的频率，从而进行特征提取工作。

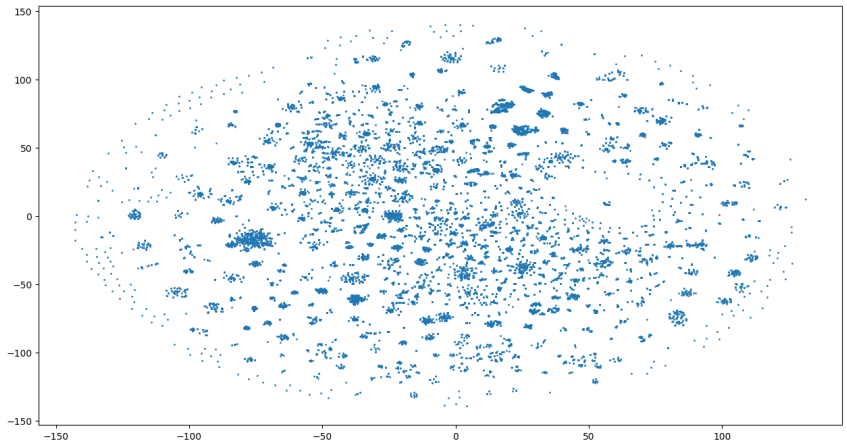

图 25: TI-IDF-TSNE2

最终计算聚类评价指标，发现结果不如只用分词模型得到的结果好。

```
{'ARI': 0.24738551498129135, 'NMI': 0.6406938987382486}
```

图 26: BERT-kmeans

结果展示

k-Means 进行了初始的数据分析和处理之后可以大致得到 BERT 特征向量提取结果和分词提取特征后的指标分布范围以及对应的降维后的数据分布。对特征向量进行了分析之后，就是对聚类方法的选择，我们首先选择的是最原始的 k-means 算法。可以发现 kmeans 算法由于是基于距离的聚类方法，所以最终的结果可视化之后可以明显看出并不能很好的将同一个大类的结果划分成一个簇，而是多个簇。

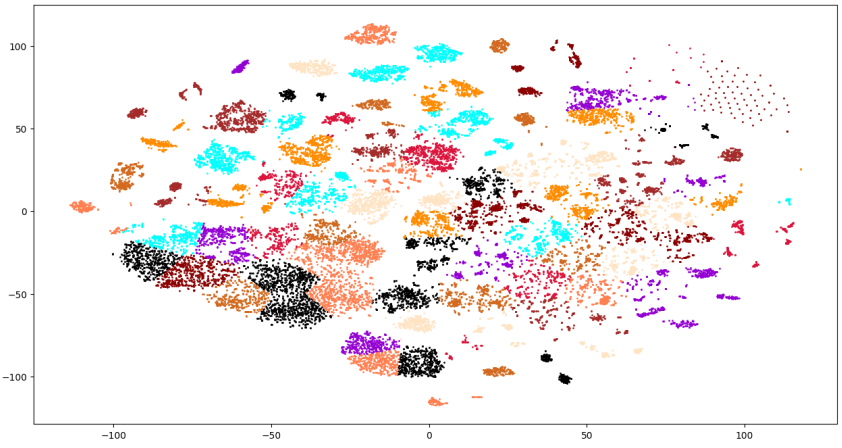

图 27: Kmeans_TSNE2_95class

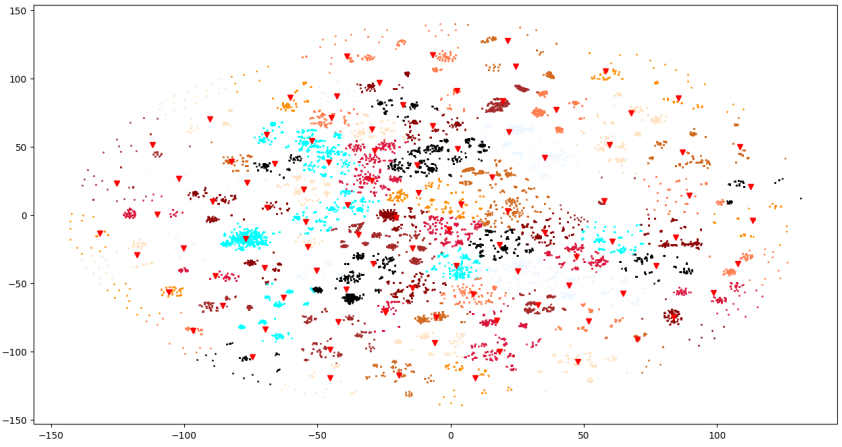

图 28: TI-IDF_kmeans_99class

CURE 之后尝试了许多其他聚类算法，但是或者运行时间过久，或者是效果不好。最终尝试了 CURE 聚类算法，能够得到一个相对不错的效果。并且 CURE 算法不受形状影响，正好符合我们数据集特征提取之后的效果。并且可以很好的适用于大规模数据，可以很好的在我们有限的算力上运行。

算法的具体步骤为

1. 从数据集中随机选取一部分适合内存大小的数据；

2. 采用传统的聚类算法 (比如层次聚类)，对样本数据进行聚类，形成初步的聚类结果;

3. 接下来一步很关键–寻找一系列的代表点 (representative points)：对于上一步聚类出来的每个组，我们在每个组内选择 k(比如 4) 个点，这四个点要尽可能尽可能尽可能地分散在这个组内，尽可能地覆盖到整个组；做法类似随机在组内选取一个离中心点最远的点，第二点要是组内所有点离第一个点最远的点，第三个点要是组内所有点离第一第二个点最远的点，以此类推；这一步的作用是尝试找到每个组的形状，定义每个组的模样；

4. 我们把每个组内的所有代表点，每一个代表点都按照固定百分比 (比如 20%) 向各自组内的中心点移动一段距离，例子如下；这一步关键在于将每个簇同比例缩小了，减少异常值的影响;

5. 接下来，我们扫描整个数据集，对于任意一点 p，我们通过计算 p 与代表点的距离，将 p 分配给最近代表点所在的簇，完成；

其中，在我们读取了一堆数据到内存后，为了提高速度，我们其实可以将这些数据分成 N 份，在每一份上面跑 CURE 算法，提取出代表点；最后再将这些代表点再使用 CURE 做一次聚类，得到最终的聚类模型，然后扫描整个数据集，把所有点分配进来。可以极大的提高运行的速度和效率。

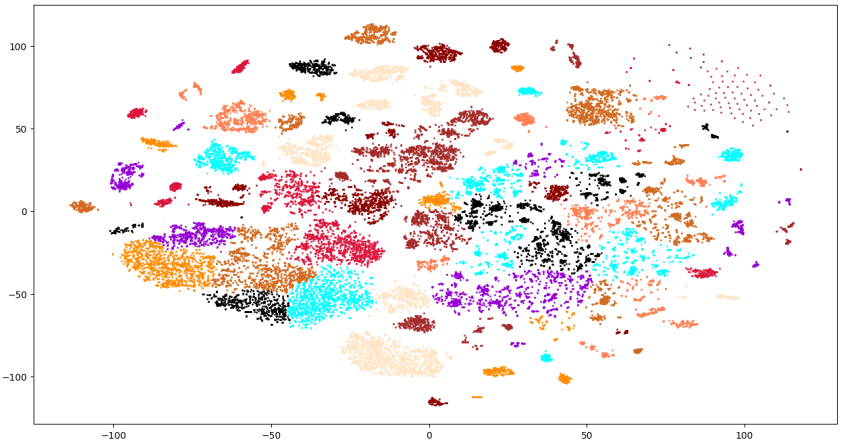

图 29: cure_TSNE2_99class

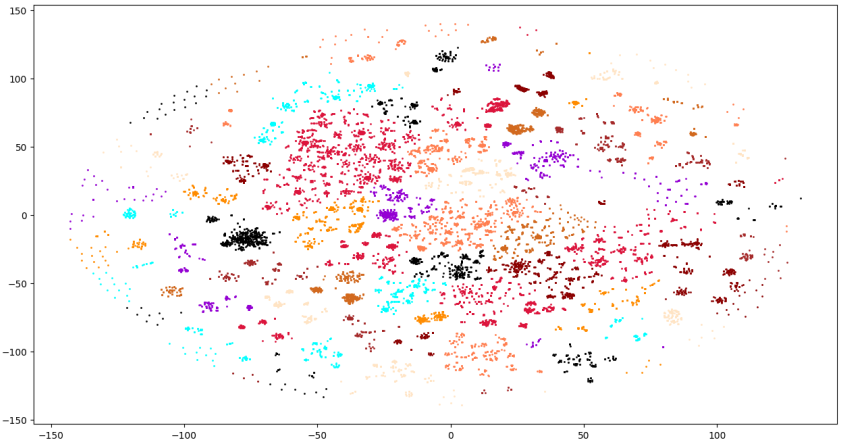

图 30: cure_99class_TI-IDF

可以明显的看出使用 TF-IDF 模型进行特征提取的效果要优于语言模型提取特征的结果，最终对 TF-IDF 提取特征后的结果使用 CURE 模型聚类分析之后，结果都优于之前的所有结果。

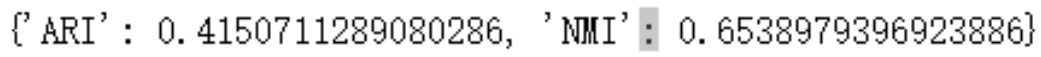

图 31: TF-IDF-cure

4. 实验总结

本次实验中，我们主要实现了对大语言模型指令数据集的聚类操作，通过对数据的特征提取，降维分析，聚类模型选择，我们最终找到了一个相对较优的聚类方法。通过本次尝试，我们体会到数据降维和模型选择在长度和维度很大的数据集处理中的重要性，以及掌握了部分处理高维数据的技巧。我们认为，对于初步获取这样的聚类结果，可以很好的区分出不同的指令数据和对话，在之后的大语言模型的数据清洗中会发挥更多的作用。

同时，本实验也有很多不足之处。例如我们只尝试了两种比较通用的语言特征提取模型，并且由于时间关系，没有进行很多的参数调整，同时降维的维度选择也是主观判断，并没有经过很系统的交叉验证。最终降维和聚类模型的选择中也是只选择了 TSNE 和 PCA、k-means 和 cure，分别两种模型，所以得到的结果十分有限。

三、结论

在本次项目中，我们主要进行了知识图谱的频繁模式挖掘和大语言模型的指令数据的聚类分析。在知识图谱的频繁模式挖掘任务中，我们尝试了使用 FPGrowth 算法对知识图谱中的数据进行挖掘，同时根据得到的频繁模式提取出一定的关联规则和有意义的元知识（概念从属关系、元路径、组合推理规则）。在实验过程中，我们主要遇到的问题为数据中的数据十分的聚集，包含了很多‘Human’相关的数据，导致频繁模式挖掘中并不能得到很多有效的信息。我们的处理方法为首先将包含 human 的数据进行人

为清洗，其次对数据支持度、置信度等阈值设置提高，可以很好的对数据进行剪枝操作。最终实现的效果为可以比较有效和快速的挖掘出数据中包含的频繁模式和一些有意义的元知识。在大语言模型指令数据集的聚类分析中，我们尝试了两种自然语言特征提取模型，分别是 TF-IDF 模型和 BERT 模型，同时在降维模型的选择中也是选择了 PCA 降维和 TSNE 降维两种方法，最终在聚类方法的选择中也是选择了 k-means 和 CURE 两种聚类算法，通过大量实验和维度参数的调整，我们最终发现使用 TF-IDF+CURE 聚类的组合方法得到的聚类效果最好。分析原因，可能是因为数据集中包含了很多特征词语，对于分词向量后聚类的方法具有天然的优势，而对于 BERT 自然语言模型则无法通过理解句子含义将句子特征很好的提取出来，所以导致最终结果的差异。