FOUR PIPELINES, ONE CORPUS · 下

我把四条本体构建路线跑了一遍

同一份语料 · 四条路线实测数据 · 结果和论文说的不一样

心晴九天 · 小红书 howru999

小红书 · 关注更新

Agent 助理 · 扫码交流

系列第 3 期（下）· 承接上期《六步流程与每步卡点》

FOUR APPROACHES, ONE CORPUS

我把四条本体构建路线跑了一遍

同一份语料 · 实测数据 · 结果和论文说的不一样

> 上期结尾那句话：前五步是把知识捞出来，第六步是确认它没错，而绝大多数方案在第五步就交货了。这期用数据验证它。

先说那张流传很广的"五个流派"图

你可能刷到过一张图，把大模型建本体的方法分成五个流派，每个派下面挂两三个项目名。我一开始也是看到这张图才去挖的。

挖完之后要说一句：**这个分法是一篇 Substack 博客自己归的，不是学术共识**，而且它把三个不同层次的东西压在了一起 — 有的是方法论文，有的是产业落地案例，有的是工程框架。

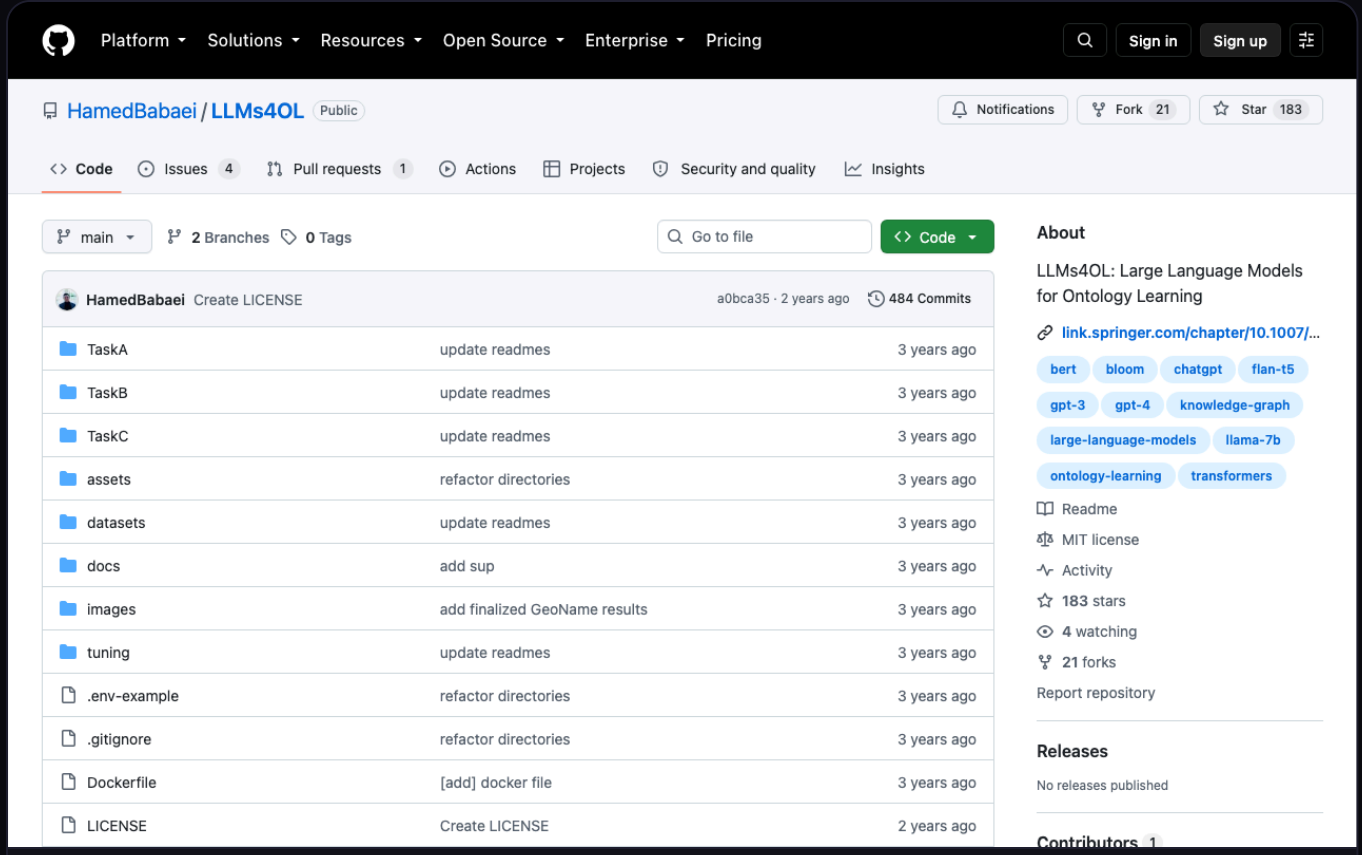
我把那十个项目一个一个核对了一遍：**只有四个真有可用的开源代码**，三个连代码库都没有只有论文，还有一个论文里写了 GitHub 地址、我点过去是 404。

那张图里没有的两个，反而更该在里面：OntoLearner（那个星最多项目的官方继任库），还有 OLLM（NeurIPS 2024 端到端微调路线）。

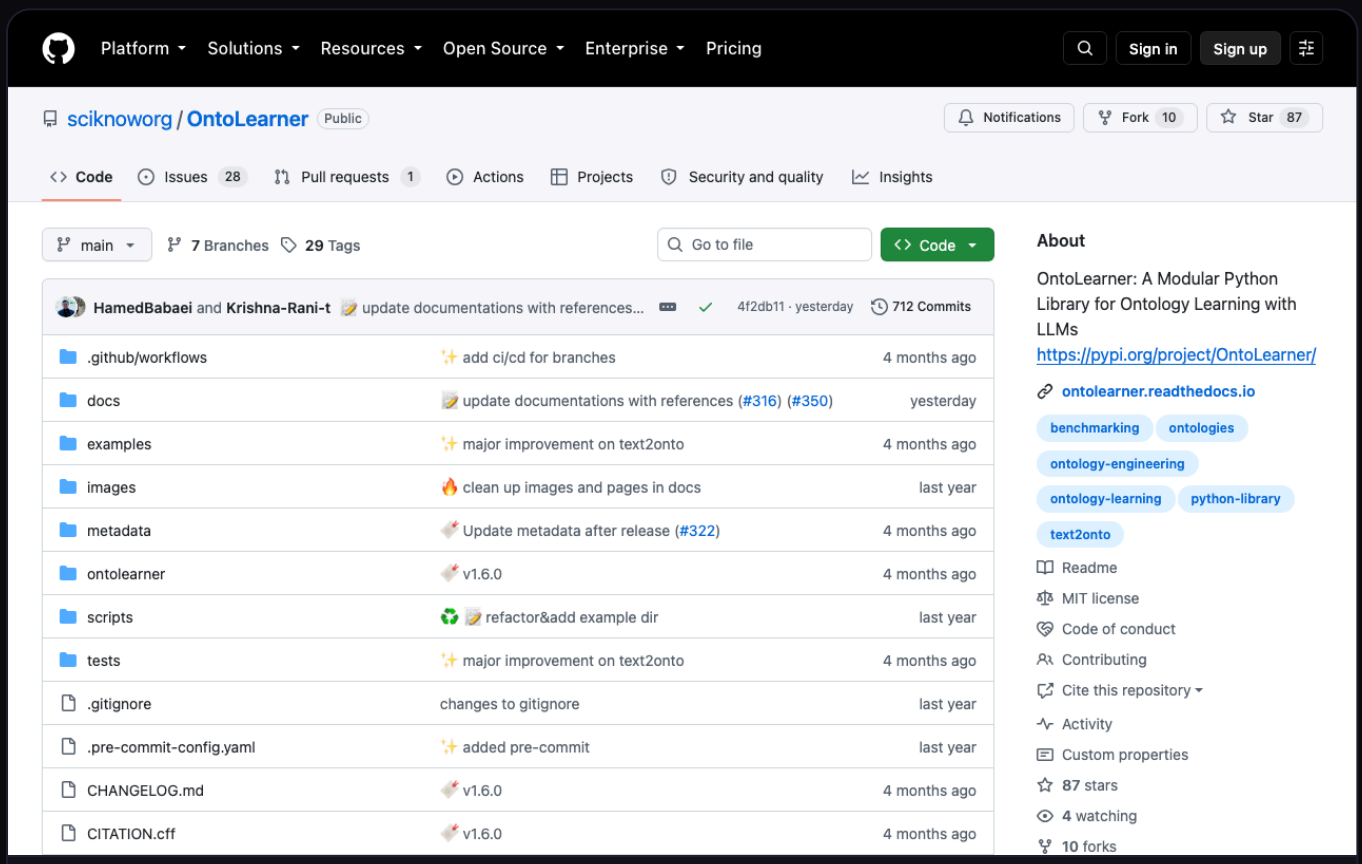


流传的那张对照表。表格本身做得挺清楚，问题在于：A 派里 Salovsky 是方法论文、IRT SystemX 是产业案例，不是同一类东西；E 派的 LLMs4OL 其实是这个方向的地基，被归到"直给派"有点委屈。另外表里 7 个项目名，只有 4 个真有可用代码

> **只有真看过提交记录才会讲的一句：星最多那个项目有 183 星，但最后一次提交是 2024 年 9 月，停更两年了。真正活着的是 OntoLearner，87 星，我查的时候它昨天还在提交。**
>
> 光看星星数，会选错。



LLMs4OL —— 183 star，是这个方向上被引最多的项目之一，但提交记录停在 2024 年 9 月



OntoLearner —— 它的官方继任库，star 数不到一半，但一直在更新。覆盖 150+ 本体源、20+ 领域

A BETTER TAXONOMY

换一个分法：schema 从哪来

不按技巧分，按一个问题分。这是真正的分歧点。

1 自由发现 — 让模型自己长出 schema

不预设任何结构，让模型从文本里长出类和关系。聚类那一派、OntoEKG、OLLM 都在这条路上。
死穴：聚类质量决定一切，且没有在线验证。

2 外部本体当护栏 — 模型只能在护栏内填

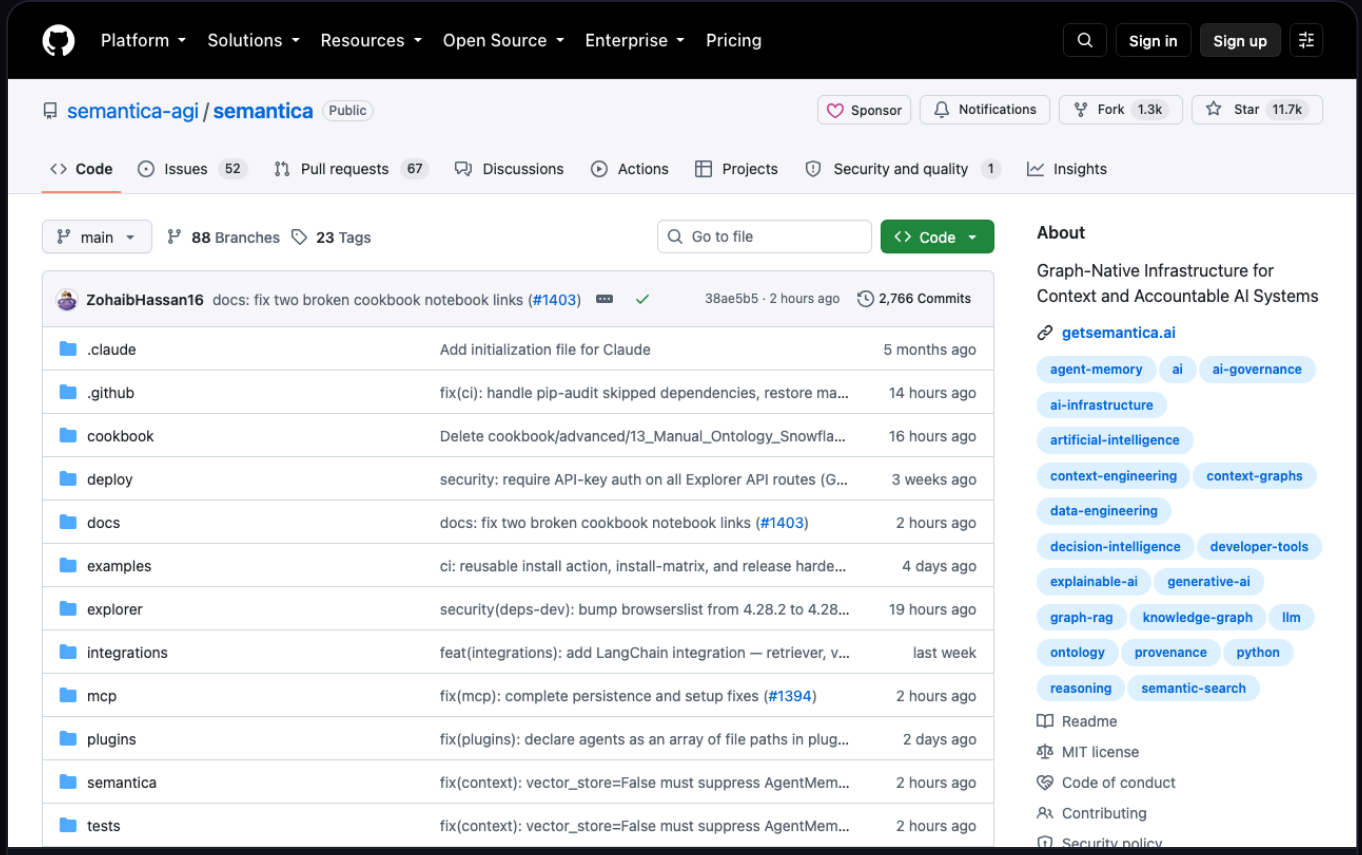
先有一个标准本体，模型只能在里面填。代表是 Wikontic（EACL 2026），它用维基数据的类型层级递归展开做约束。
死穴：中文垂直领域的概念在维基数据里极稀疏 — "小规模纳税人""进项税额转出"基本没有对应项。

3 schema 先行 — 先设计再灌数据

人加模型先把 schema 设计出来，再往里灌数据。代表是 OntoKG（arXiv:2604.02618），把维基数据属性分成 94 个模块。
它自己在论文里承认：产出的是一个声明式 schema，不是形式化的本体。

4 不用大模型 — 从数据结构确定性推导

从数据自身的结构确定性地推出类和属性，再用 SHACL 校验。代表是 Semantica — **11,644 星，MIT 协议，活跃更新**，自称"给 AI Agent 的开源 Palantir"。



Semantica —— 11.7k star、1.3k fork、MIT 协议、2766 commits。侧栏标签里同时挂着 ontology、knowledge-graph、context-engineering、provenance

> **Semantica 的 README 里写着：本体构建、推理、溯源，都不需要大模型。**

这就是我说的分裂

学术界那几派全挤在第一条路上比精度，而 GitHub 上最火的那个开源本体基础设施 — 比前面所有项目加起来的三十倍还多 — **直接把大模型从本体构建环节踢出去了。**

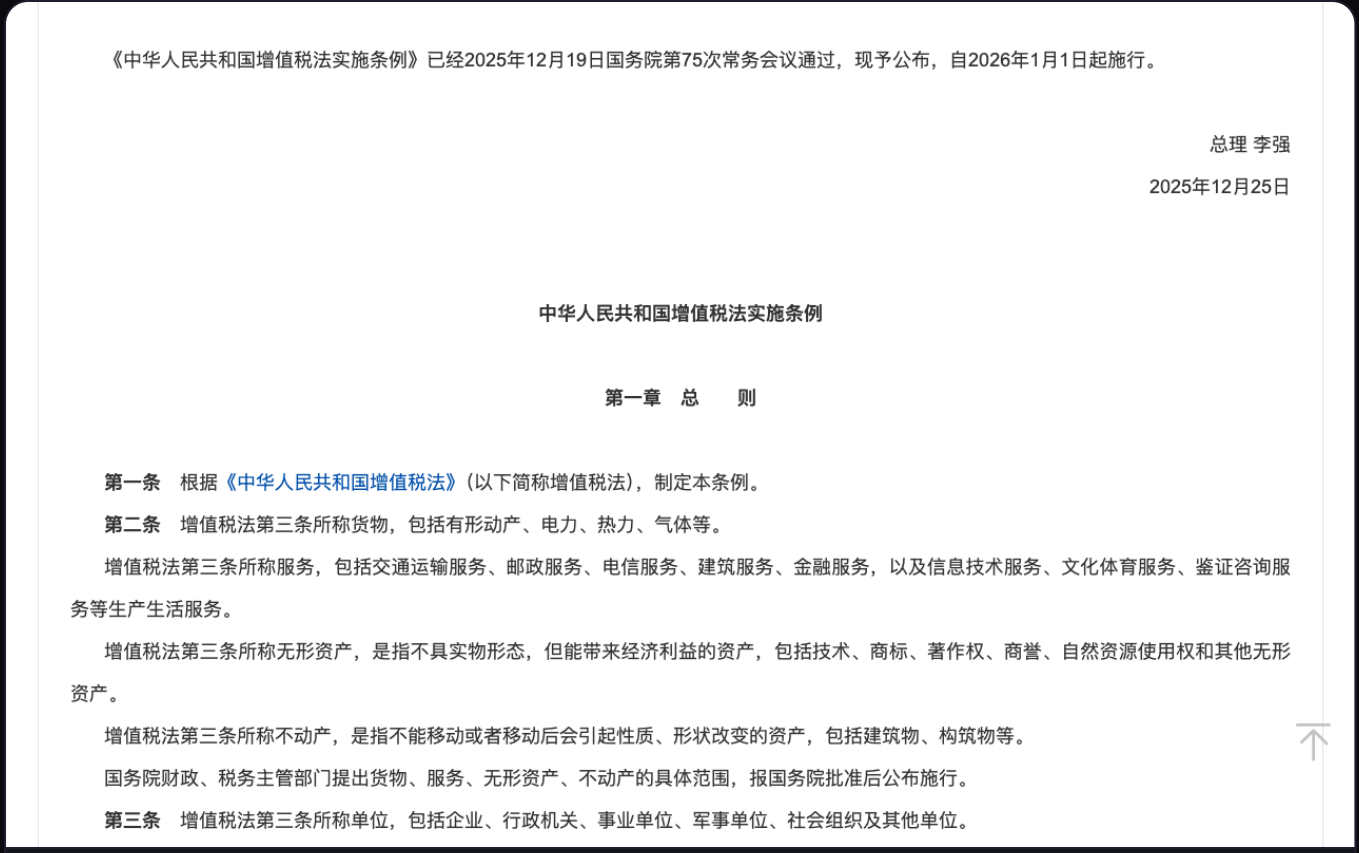
THE CORPUS

先说语料，这一步很关键

增值税进项抵扣 · 国家税务总局 / 上海市税务局公开法规原文

14,621 字符

增值税法 + 实施条例 + 进项税额抵扣公告 · **不含任何客户数据**



《中华人民共和国增值税法实施条例》已经2025年12月19日国务院第75次常务会议通过，现予公布，自2026年1月1日起施行。

总理 李强

2025年12月25日

中华人民共和国增值税法实施条例

第一章 总 则

第一条 根据《中华人民共和国增值税法》（以下简称增值税法），制定本条例。

第二条 增值税法第三条所称货物，包括有形动产、电力、热力、气体等。

增值税法第三条所称服务，包括交通运输服务、邮政服务、电信服务、建筑服务、金融服务，以及信息技术服务、文化体育服务、鉴证咨询服务等生产生活服务。

增值税法第三条所称无形资产，是指不具实物形态，但能带来经济利益的资产，包括技术、商标、著作权、商誉、自然资源使用权和其他无形资产。

增值税法第三条所称不动产，是指不能移动或者移动后会引起性质、形状改变的资产，包括建筑物、构筑物等。

国务院财政、税务主管部门提出货物、服务、无形资产、不动产的具体范围，报国务院批准后公布施行。

第三条 增值税法第三条所称单位，包括企业、行政机关、事业单位、军事单位、社会组织及其他单位。

语料来源之一：《中华人民共和国增值税法实施条例》，国务院令第826号，2025 年 12 月 19 日国务院常务会议通过。国家税务总局政策法规库

为什么选这个切口

- 《增值税法》2026 年 1 月 1 日才施行
- 《实施条例》（国务院令第826号）2025 年 12 月 19 日才通过
- 模型训练数据里几乎没有这部法
- 它不能靠记忆蒙，必须真从语料里抽

为什么这点重要

- 很多本体 demo 用英文、成熟领域的语料
- 模型早就背过，跑出来效果很漂亮
- 但测的是记忆不是能力
- 换到自己公司的场景就废了

HEAD-TO-HEAD

四臂实测

同一份语料，四条路线各跑一遍。

	OntoLearner	Semantica 确定性	Semantica LLM	带验证闭环
路线	判定式	结构推导	生成式	生成+闭环
类	不产出	10 (全叫 Entity)	46	**125**
规则	—	**0**	**0**	**74**
例外	—	**0**	**0**	**24**
三元组	—	—	—	**1,171**
验证	无	未触发	无	**语法+双推理机+结构**

> **核心观察：前三条路都不产出规则和例外。这不是配置问题 — 那个框架的 prompt 只要求 classes 和 properties 两项，数据结构里根本没有放规则和例外的位置。**

第一个：不用大模型那条路，第一步就断了

我喂给它十个中文实体，它产出十个类 — 但名字全都叫 `Entity`，而且共用同一个 URI。

我去定位了根因，找到它的命名规范化函数，拿中英文分别喂：

中文输入

纳税人 → **Entity**

小规模纳税人 → **Entity**

一般纳税人 → **Entity**

进项税额 → **Entity**

销项税额 → **Entity**

无形资产 → **Entity**

6 个输入 → **1 种输出**

英文输入

taxpayer → **Taxpayer**

small scale taxpayer → **SmallScaleTaxpayer**

input tax → **InputTax**

output tax → **OutputTax**

4 个输入 → **4 种输出，全对**

```
$ python arm-b-semantica/run_arm_b.py

[corpus] 14697 chars from 3 files

=== 根因探针：命名规范化 ===
{
  "chinese": {
    "纳税人":       "Entity",
    "小规模纳税人":  "Entity",
    "一般纳税人":    "Entity",
    "进项税额":     "Entity",
    "销项税额":     "Entity",
    "无形资产":     "Entity"
  },
  "english": {
    "taxpayer":             "Taxpayer",
    "small scale taxpayer": "SmallScaleTaxpayer",
    "input tax":            "InputTax",
    "output tax":           "OutputTax"
  },
  "chinese_distinct_outputs": 1,
  "english_distinct_outputs": 4,
  "verdict": "中文全部折成同一个类名，英文保持区分"
}
```

实际跑出来的探针输出 —— 6 个中文词全部折成 Entity，4 个英文词全对。这段是可复现的，不是我手写的对照表

> **我讲这条不是为了踩它 — 它的推理引擎、双图存储、时序逻辑设计都很扎实。我讲的是另一件事：开源基础设施的成熟度是分语言的。你在中文场景里，不能照搬星星数最高的方案。**

第二个：同一框架换成大模型抽 — 名词抄下来了，规则丢了

46 个类，26 个属性，层级也对。看起来不错。但我拿第二十二条去审计它 — 那一条列了五类不得抵扣的进项税额：



第十五条 境外单位和个人在境内发生应税交易，以购买方为扣缴义务人；按照国务院的规定委托境内代理人申报缴纳税款的除外。

扣缴义务人依照本法规定代扣代缴税款的，按照销售额乘以税率计算应扣缴税额。

第十六条 销项税额，是指纳税人发生应税交易，按照销售额乘以本法规定的税率计算的增值税税额。

进项税额，是指纳税人购进货物、服务、无形资产、不动产支付或者负担的增值税税额。

纳税人应当凭法律、行政法规或者国务院规定的增值税扣税凭证从销项税额中抵扣进项税额。

第十七条 销售额，是指纳税人发生应税交易取得的与之相关的价款，包括货币和非货币形式的经济利益对应的全部价款，不包括按照一般计税方法计算的销项税额和按照简易计税方法计算的应纳税额。

第十八条 销售额以人民币计算。纳税人以人民币以外的货币结算销售额的，应当折合成人民币计算。

第十九条 发生本法第五条规定的视同应税交易以及销售额为非货币形式的，纳税人应当按照市场价格确定销售额。

第二十条 销售额明显偏低或者偏高且无正当理由的，税务机关可以依照《中华人民共和国税收征收管理法》和有关行政法规的规定核定销售额。

第二十一条 当期进项税额大于当期销项税额的部分，纳税人可以按照国务院的规定选择结转下期继续抵扣或者申请退还。

第二十二条 纳税人的下列进项税额不得从其销项税额中抵扣：

（一）适用简易计税方法计税项目对应的进项税额；

（二）免征增值税项目对应的进项税额；

（三）非正常损失项目对应的进项税额；

（四）购进并用于集体福利或者个人消费的货物、服务、无形资产、不动产对应的进项税额；

（五）购进并直接用于消费的餐饮服务、居民日常服务和娱乐服务对应的进项税额；

（六）国务院规定的其他进项税额。

第二十二条原文六项。第（五）项那句"购进并**直接用于消费**的餐饮服务"，是后面所有对比的关键

✓	（一）简易计税方法计税项目	建了
✓	（二）免征增值税项目	建了
⚠	（三）非正常损失项目	译成 NonPermanentLoss — 语义错了
✗	（四）集体福利或个人消费	完全没有
✗	（五）餐饮、居民日常、娱乐服务	完全没有

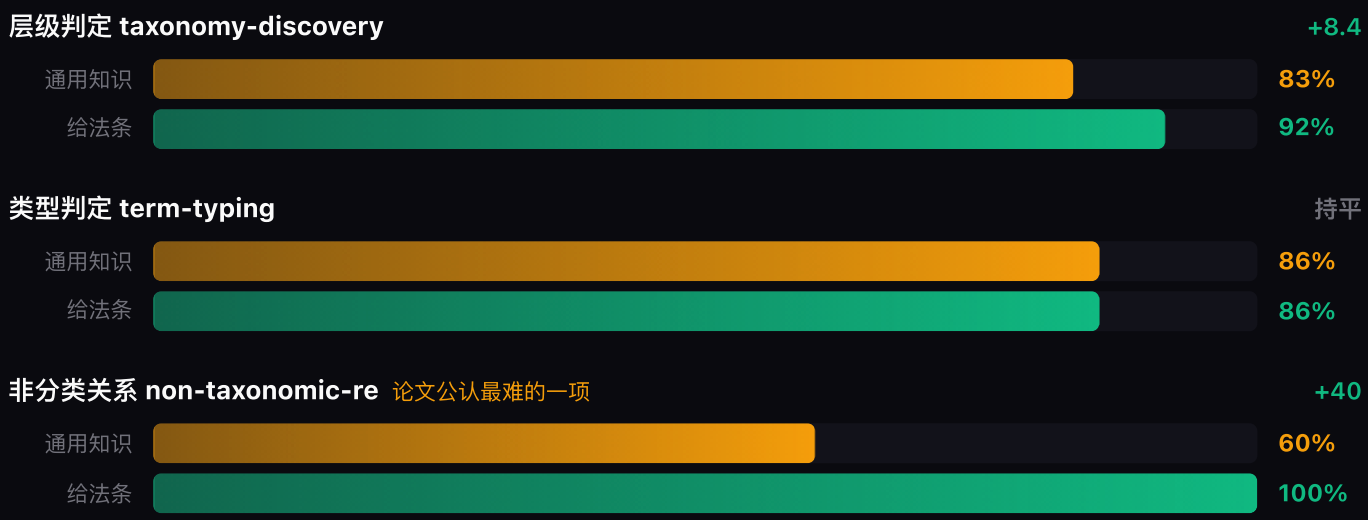
"非正常损失"指的是被盗、霉烂、违法损毁，跟"永久不永久"没关系。五类里覆盖三类，其中一类还译错了。

> **更关键的是：整个本体里没有任何一个地方表达了"不得抵扣"这四个字。没有对应的类，没有对应的属性。**

第三个：一句 prompt 推翻一个两年的结论

OntoLearner 是判定式的：给一对概念，回答 yes 或 no。我建了 24 条测试集，每条都写了法条出处，可逐条核对 — 有正例、有方向故意颠倒的反例、还埋了几个陷阱。

我读它的 prompt，发现里面有一句 `Use general world knowledge` 。通用知识里没有这部今年才施行的法啊。所以我**只改了这一句**，换成"只依据下面给出的法条原文"并把法条附上。别的一个字都没动。




```
$ python arm-a-ontolearner/run_inference.py

[A2 frontier] model=claude-sonnet-5, rounds=3

=== A2 稳定性（3 轮同 prompt 同模型）===
  每轮准确率：[0.792, 0.792, 0.792]
  答案摇摆的题数：0 / 24

[A3 grounded] corpus=14621 chars

============================================================
A2（通用知识） vs A3（给法条）
============================================================
  taxonomy-discovery      0.833  ->  0.917
  term-typing             0.857  ->  0.857
  non-taxonomic-re        0.6    ->  1.0   <-- 论文公认最难的一项

  A3 仍错 ✗ {'parent': '应税交易', 'child': '视同应税交易'} gold=True pred=False
  A3 仍错 ✗ {'term': '餐饮服务', 'type': '不得抵扣情形'} gold=False pred=True
                ↑ 漏掉"购进并直接用于消费"这个限定条件 —— 这就是例外
```

实测输出。上半是稳定性（3 轮完全一致、0 题摇摆），下半是 A2 vs A3 对照。最后两行是给了法条仍然答错的两条 —— 第二条就是漏掉限定条件的那个例外

> **论文里说了两年、公认最难的那一项，满分。**
>
> 不是模型不会抽非分类关系，是没人把材料给它。上期那句"它不在词里，它在制度文件里"，讲的就是这件事。

怕是随机波动，同 prompt 同模型跑了三轮：准确率完全一样，24 道题**一道都没摇摆**。判定是稳定的。

但有两条题，给了法条它还是错 — 这两条最有意思

第一条：视同应税交易，是不是应税交易的子类？ 法条第五条明明白白写着"视同应税交易"，它答 no。

这不算它错。它的理由是法律拟制不构成真正的 is-a — 这在本体论上是个真实的分歧，不是知识错误。**这种题得有人来裁定**。本体建模里有一类问题，不是查资料能解决的，是要有人拍板的。

第二条：餐饮服务，是不是不得抵扣的情形？ 它答 yes。正确答案是 no。

因为法条写的是"**购进并直接用于消费的**餐饮服务"不得抵扣。你买了转手卖给下游客户，是可以抵扣的。这是财税从业者的基本功，而它漏掉了那个限定条件。

这条就是例外。而例外，填不进一个只能回答 yes 和 no 的框架里。

第四个：加上验证闭环之后

思路抄 NeOn-GPT：抽取完不算完 — 查语法、查逻辑一致性、查结构缺陷，把报错拼成诊断报告喂回去重修。但我加了两样它没有的：**每条规则必须写法条依据和验收口径；必须产出例外表，一条例外都没有直接判不合格。**

	类	规则	例外	报错
第1轮	**87**	**39**	**18**	1 err / 2 warn
第2轮	**87**	**41**	**19**	1 err / 1 warn
第3轮	**87**	**41**	**19**	1 err / **0 warn**
↓ 补齐缺失范围后重跑 ↓				
一轮	**125**	**74**	**24**	**0 err** · 收敛

```
$ python arm-c-studio/final_merge.py

  + core:          +376 → 376
  + deduction:     +330 → 706
  + preference:    +194 → 900
  + invoice_fixed: +271 → 1171
[merged] 1171 三元组，四段全覆盖

--- 第 1 轮验证 ---
  统计：{'classes': 125, 'object_properties': 28,
         'datatype_properties': 46, 'rules': 74,
         'exceptions': 24, 'triples': 1171}
  error 0 / warning 5
    ~ (pitfall/P04-isolated-class) [vat:SalesAllowanceSuspensionOrReturn]
    ~ (pitfall/P04-isolated-class) [vat:ConsignedExportProcedure]
    ~ (pitfall/P04-isolated-class) [vat:AnnualSettlement]
    ~ (pitfall/P04-isolated-class) [vat:WithholdingObligationTime]
  ✓ 无 error，闭环收敛

============================================================
完整本体闭环轨迹
============================================================
  第1轮  类125  规则 74  例外 24  error  0  warning  5

  采用：类125 规则74 例外24 三元组1171 error 0
```

四段合并后的实际运行日志 —— 1171 三元组、125 类、74 规则、24 例外，一轮 error 0 收敛。剩下 5 条 warning 全是孤立类，不影响逻辑

74 / 74 条规则全部带法条依据和验收口径。**24 / 24 条例外**全部挂到具体规则上、全部有限定条件。第二十二条五类不得抵扣，**这次五类全覆盖**。

第（六）项"国务院规定的其他"没建 — 那是空白授权条款，没有实体内容，不建才是对的。

最关键的是我去翻那 24 条例外，其中一条是这么写的：

EXCEPTION #12 — 自动生成，非人工编写

餐饮服务转售不适用不得抵扣的例外

第二十二条第（五）项"不得抵扣"的适用前提是"购进并直接用于消费"，即购进方本身消耗使用该服务；若纳税人购进餐饮服务、居民日常服务或娱乐服务后系转售，则不适用该规则。

限制规则：购进并直接用于消费的餐饮、居民日常服务和娱乐服务对应进项税额不得抵扣规则

arm-c-studio/out/final_full.ttl · 24 条例外之一 · 自动生成，非人工编写

```
vat:Exception_CateringOnlyWhenDirectConsumption a vat:Exception ;
    rdfs:label "餐饮服务转售不适用不得抵扣的例外"@zh ;
    vat:exceptsRule vat:Rule_NonDeductible_Item5_CateringResidentEntertainment ;
    vat:exceptionCondition "第二十二条第（五）项'不得抵扣'的适用前提是'购进并直接用于消费'，即购进方本身消耗使用该服务；若纳税人购进餐饮服务、居民日常服务或娱乐服务后系转售（例如作为经纪代理、打包销售给下游客户等而非自行消费），不属于'购进并直接用于消费'，则不适用本条不得抵扣的规定，对应进项税额可以正常抵扣。"@zh ;
```

产物里这条例外的原始 Turtle。注意它自己补出了"经纪代理、打包销售给下游客户"这些法条里没直接写的转售场景

> **就是 OntoLearner 答错的那一条。它不但建出来了，而且不是照抄法条 — 它把那个限定条件识别成了规则的适用边界，还自己补出了转售这个场景。**

还有一些前面几个方案完全没有的：资产重组不视为应税交易、原值不超过 500 万元的长期资产不用逐年调整、国内旅客运输凭非专用发票怎么按公式算。

FOUR FINDINGS

过程里的四个发现

这不是一个成功案例展示。这四条比结果本身更有用。

1 我一开始追了一个假问题

前两段跑完，闭环剩一条一致性错误，怎么都消不掉。

我做过的定位尝试（全部无果）

查 disjoint 冲突 · 查属性同时是 Object 和 Datatype · 查值域指向类 · 查同一属性多个不相容值域 · 查值域与实际取值冲突 · 逐条剥离 45 条值域声明 · 逐属性整体剥离 45 个属性。

按 predicate 分组剥离显示：去掉全部 `rdfs:range`（45 条）*或*去掉全部 `rdf:type`（192 条）都能变一致 — 当时的结论是"多条公理交互产生的矛盾"。

真相

后来我把剩下的段补齐了，重跑一遍，**这条错误自己没了，一轮就收敛**。真正的原因是本体残缺 — 缺掉的那部分内容，让剩下的公理没法自圆其说。

教训

在本体还不完整的时候纠缠一致性报错，很可能是在追一个假问题。先把范围补齐，再看还剩什么。

顺带暴露的问题

推理机一旦判定本体不一致，就**拒绝回答任何后续问题** — 你问它哪个类有问题，它直接抛 `InconsistentOntologyException`。验证器能告诉你"错了"，但不告诉你"错在哪"。

2 两个推理机对同一个本体给出相反结论

HermiT 说一致，Pellet 说不一致。而本体本身没错。

根因

有四个属性声明了值域是 `rdf:langString`，值也确实带着 `@zh` 语言标记 — **语义上完全正确**，但 Pellet 对这种显式写法的处理更严格。

后果很实际

我第一次跑的时候只信了一个推理机。它报不一致，我就把报错喂回去让模型修。模型不知道该改哪，就开始**删规则 — 从 15 条被删到 9 条**。

它是在用"删掉内容"的方式，让报错消失。

解法

改成两个推理机交叉验证，**只有两个都说不一致才算真错**。另外把当前规则数和例外数写进修复指令，要求修完不得少于这个数。

给企业方的提醒

一个只报"有问题"但不说"问题在哪"的验证器，**会诱导你的团队用错误的方式消除报错**。而这种错误，比不做验证更难发现。

3 解决截断的办法不是加大输出上限

大领域本体没法一次生成完，必须分段。这里有三个坑。

坑一：加大上限反而更糟

我一开始把上限从 16,000 调到 32,000 — 结果触发了另一个限制（超长请求被拒），更糟。**真正有效的是反过来：裁剪输入、拆分目标。**只喂相关章节、把一段再拆两半，就跑通了。

坑二：空产物会被误判成通过

输出被截断 → 提取到空串 → 空本体自然零报错 → **闭环误判"收敛"**。我踩过一次。验证器必须显式检查"有没有类"。

坑三：截断的产物可以救

按空行做块级回退，从末尾一块块往回退到能完整解析的位置。实测救回 306 条里的 300，只丢最后一条不完整的规则。

还有一个环境坑

我用的端点默认开启扩展思考。实测 16,000 tokens **全被 thinking 吃掉**（thinking_tokens=15,999），正文一个字都没有 — 必须显式关掉。这个坑排查了很久。

4 修复要用补丁，不要整体重写

本体上了几百条三元组之后，让模型重新输出全文一定会被截断。

现象

706 条三元组的本体，一要求整体重写就超输出上限被截断，**连正确的部分一起丢掉**。

解法

改成只输出"要删哪几条、要加哪几条"两个补丁块，在图层面应用而不是文本层面。改完之后规则和例外**单调增长，不再倒退**。

TAKEAWAY

收尾

四条路跑完，它们缺的是同样三样东西 — **例外、动作、验收标准**。上期讲过这三样，这期的数据都在证明它。

> **这里要强调一句，因为很容易被误读成"我的方案更好"：最后那个本体用的就是普通的商用模型，跟前面几个方案没有模型上的差别。差别只在于，我在抽取时明确要求它产出规则和例外，并在验证环节规定"一条例外都没有就判不合格"。**

你要什么，才可能得到什么。

那个 46 个类零条规则的结果，不是模型不行，
是那个框架的数据结构里就没有规则和例外的位置。

而这三样填不了，根本上不是因为算法不够好。
是因为它们不在算法里，在人的脑子里。

立项判断：四条标准

四条同时满足，值得做；缺一条，先别急着立项。

1. **高频重复，且直接影响收入、成本、履约或风险**
   一年发生三次的事，不要上系统。

2. **输入材料能数字化，且能连到必要的数据与系统**
   材料在别人手里、系统连不上，本体建得再漂亮也跑不起来。

3. **有愿意共同定义规则和例外的专家，且有明确经济买方**
   这两个可以不是同一个人，但都必须存在。没有专家，例外没人能给；没有买方，做完了没人签字。

4. **结果能对上人工基线，验收口径双方事先能约定**
   事后再谈标准，一定谈不成。

> 上期那张自检清单我把这四条也加进去了，还补了这期跑出来的几个坑：推理机怎么交叉验证、报错回灌时怎么防止模型偷偷删规则、分段抽取的三个坑。需要的伙伴在评论区说一声。

下一期：让 agent 拿本体做动作

先回应上期评论区一个很好的问题

有位伙伴问：现在很多团队都在用多 agent 编排的方案落地，并没有用 ontology，你怎么看。

我的回答是这两件事不冲突。**agent 编排解决"怎么干"，ontology 解决"按什么干"** — 前者是引擎，后者是地图。规则简单的时候确实用不上，一旦例外多、要留痕、要能验收，规则总得有个地方存。

但这个回答只答了一半。因为今天讲的这一整套，产出的还是**一张只读的图** — 它能回答问题，不能改变业务。而企业买的是能动手的系统。

所以下一期讲第三样缺的东西。会拆成三个问题：

1. **动作怎么定义才算一等公民**
   而不是在提示词里塞几句"你可以调用这些工具"。动作要有前置条件、后置状态、失败回滚 —— 这些得写在本体里，不是写在 prompt 里。

2. **授权、留痕、撤回怎么落在数据结构上**
   这是业务方敢不敢让系统跑在真实流程上的分界线。三件缺一个，方案就停在演示阶段。

3. **模型判断错了，能不能在动作执行前挡住**
   今天的验证闭环是事后查本体对不对。下一步是事前查这个动作该不该做 —— 本体和推理机能不能当刹车，而不只是当地图。

我还是用今天这套增值税本体接着往下做，不换语料。这样你能看到同一个东西从只读变成能动手，中间到底加了什么。

如果你手上正好有类似场景，或者今天讲的哪一步你踩过不一样的坑 — 评论区说一声，我下期尽量带进去一起讲。

系列第 3 期（下）· 四条路线实测

语料：国家税务总局 / 上海市税务局公开法规原文 · 不含任何客户数据

环境：Python 3.12 · semantica 0.6.7 · OntoLearner 1.6.0 · owlready2 0.51 · HermiT + Pellet (openjdk 26)