《大语言模型与应用实践-2025》课程设计项目技术报告（CVPR风格撰写）

摘要（Abstract）

长文本领域翻译的主要难点在于：跨章节术语与实体的一致性、领域风格的稳定保持、以及面向出版标准的可验证质量控制。本文提出一套**基于多智能体工作流（Agent Workflow）**的长文本翻译系统，采用 LangGraph 将翻译过程拆解为可控节点，并引入（1）**术语/实体智能体**：自动抽取并查证术语，构建全局术语表；（2）**翻译与润色智能体**：结合风格元数据、术语表与历史上下文生成译文；（3）**评估与自我修正智能体**：通过回译一致性与多维质量评估触发迭代修正；（4）**章节级人机协作**：在关键质量控制点引入人工审查并对全局一致性进行落盘。系统在计算机视觉（CV）经典论文长文本翻译场景中进行验证，报告将给出流程、实现细节、实验设置与消融设计；**结果部分按要求占位，待全部跑通后补充。**

1. 引言（Introduction）

1.1 背景与问题定义

大语言模型（LLM）在短文本翻译中表现优异，但在长篇书籍/论文（章节级、跨段落依赖）翻译时，容易出现：

- **上下文丢失（Context Loss）**：章节跨度导致风格、译名策略漂移。
- **术语不一致（Inconsistency）**：同一术语多译（例如 “Dropout” 被译为“丢弃法/随机失活/保留原文”）。
- **幻觉与漏译（Hallucination & Omission）**：长难句结构复杂导致语义偏差与信息缺失。

为此，我们将任务定义为：给定结构化长文本 $X=\{x_{c,k}\}$（*章节 c 的第 k 个 chunk*），*在满足风格一致、术语一致与语义准确的约束下，生成译文* $Y=\{y_{c,k}\}$，并提供可追溯的质量评估与迭代修正记录。

1.2 方法概览与贡献

本文系统基于 `try/core/graph.py` 与 `try/core/nodes.py` 实现，主要贡献包括：

- **工作流化翻译范式**：使用 LangGraph 将翻译拆解为“风格分析→术语挖掘→RAG查证→翻译→评估→修正→持久化”的可控节点，并显式建模状态 `TranslationState`。
- **全局一致性机制**：通过章节级术语汇总与全局术语表注入，在 Prompt 层强制统一译名，并在输出侧持久化/回写。
- **TEaR式可验证闭环**：引入回译（Back-translation）与结构化质量评估（QualityReview），以条件路由 `quality_gate` 驱动迭代修正。
- **可运行与可复现工程实现**：落盘 `chunk_*.json`、记录 `refinement_history`，并加入自动模式下的速率限制（避免 API 触发限流）。

2. 项目范围与数据构建（Data Construction）

2.1 领域与源文本

本项目选择**计算机视觉（CV）**领域经典论文作为“领域书籍”实例：

- **AlexNet**: *ImageNet Classification with Deep Convolutional Neural Networks* (2012)
- **VGG**: *Very Deep Convolutional Networks for Large-Scale Image Recognition* (2014)

- **ResNet**: *Deep Residual Learning for Image Recognition* (2016)

其特点是：术语密度高、论证结构严谨，适合作为长文本领域翻译系统的验证基准。

2.2 结构化清洗与切分策略

系统将源文本预处理为 JSON（保留章节结构，如 Abstract/Introduction/Method）。随后执行两级切分：

- **章节切分**：`split_epub_by_chapter(...)` 生成 `chapters=[{title, content}]`。
- **chunk切分**：`split_chapter_into_chunks(content)` 将章节内容切分为逻辑块，控制每块长度以兼顾上下文充分性与上下文窗口限制。

2.3 翻译单元与元数据

每个翻译单元携带最小可追溯元数据：`book_id/chapter_id/chunk_id/thread_id`，并可附加：

- **global_glossary**：全局术语表（跨章节）。
- **critique/is_retry**：章节级审查或重译反馈。

2.4 RAG（翻译记忆库）的更新与统计

为降低"同一术语多译"与"跨章节漂移"，系统维护并持续更新翻译记忆库（Translation Memory）。本次实验使用的更新后 RAG 备份为 `try/output/rag_backups/rag_backup_20260112210224.json`。该 RAG 的关键特点是：其中包含**人工审查/人工修改过的术语条目**（在字段层面可追溯），这些条目会在检索时作为"证据"被注入提示词，从而影响术语规范化与最终译文。

需要强调的是：RAG 的增强并不一定单调提升所有指标。它同时带来一个典型权衡：**召回更强**（更多可用"证据"）与 **噪声更大**（同义译法/风格差异更容易被检索到）并存；因此在缺少人工校准的情况下，可能出现术语指标波动，进而影响综合评价（见 5.4.1）。

3. 方法（Method）

3.1 系统总体架构（Workflow Graph）

系统采用"线性流水线 + 质量闭环"的总体架构：先完成风格识别、术语/实体抽取与查证，再进行翻译生成；随后通过回译与质量评估对译文进行门控，若未达标则触发针对性修正并重新评估，直至达标或达到最大迭代次数。该设计的关键在于把"生成"和"验证"解耦，并把验证结果结构化地反馈给修正阶段，从而形成可控的迭代优化过程。

具体而言：

- **前向流水线（一次性）**：风格/语域识别 → 术语候选挖掘 → 基于外部记忆的术语查证与规范化 → 约束条件下注入式翻译生成。
- **质量闭环（可迭代）**：回译一致性检查 + 多维质量评估 → 质量门控（通过/修正/强制停止） → 针对性修正 → 重新评估。

> **图 1（占位）**：系统工作流示意图（节点与状态流转）。

3.2 统一状态表示（TranslationState）

系统在节点间传递的是一个"可追溯的统一状态"（state），它同时承载：

- **输入与定位信息**：用于把长文本拆成可处理单元，并在输出侧可回收/复现（例如 chapter/chunk 的索引与线程标识）。
- **可复用上下文**：包括跨章节术语表、章节记忆、相似翻译示例等，用于稳定风格与一致性。
- **中间产物**：如风格元数据、术语候选、规范化术语表等，作为后续生成的约束条件。
- **质量证据链**：回译文本、质量分数、错误类型、具体问题与改进建议，以及每次迭代的记录。

与"仅保存最终译文"的系统不同，这种状态设计强调**可审计性**：任何一个 chunk 的译文都能追溯到当时的上下文、约束条件与评估反馈，为后续质量分析、错误归因与人工复核提供依据。

3.3 Agent 1：术语与实体智能体（Terminology & Entity Agent）

该智能体负责把"术语一致性问题"从生成阶段提前为一个显式子任务：先识别需要统一的语言单位，再利用外部知识/记忆进行查证与规范化，最终形成可约束生成的结构化术语表。

3.3.1 术语挖掘（Term Mining）

系统通过提示词引导模型做"难词/关键单位抽取"，并使用结构化输出约束抽取结果（例如 NER、领域术语、缩写、可能存在文化负载的表达）。关键约束：

- 仅输出英文原文词汇（不在该阶段翻译）。
- 对人名提示"可识别但翻译阶段必须保留原文"。

3.3.2 RAG查证与规范化（Retrieval & Normalization）

对每个候选术语，系统执行"检索—对齐—规范化"三步：

- **检索（Retrieve）**：从翻译记忆库中召回与该术语最相关的历史对齐片段（top-k）。当前实现基于 Elasticsearch 的 `multi_match`，并对英文原文给予更高权重（`en^2`），辅以 `fuzziness="AUTO"` 以提高鲁棒性。
- **对齐（Align）**：将检索结果整理为可直接注入提示词的对齐证据（例如 "英文 → 中文（可选上下文）"）。
- **规范化（Normalize）**：模型基于"检索证据 + 当前句子语境"输出结构化术语条目（包含译名、类型、语境含义与理由），并遵循强约束规则。

其中检索策略要点为：

- Elasticsearch `multi_match` 字段权重：`en^2, zh, title^0.5`
- `fuzziness="AUTO"` 以容忍拼写差异
- 返回可直接拼入 Prompt 的对齐片段 `- en → zh (context)`

规范化时，系统强制执行规则（写入提示词约束）：

- **人名/作者名**：`suggested_trans` 必须保留英文原文。
- **缩写/专名**：可保留原文并补中文解释（视上下文）。

3.4 Agent 2：翻译与润色智能体（Translation & Refinement Agent）

该智能体在“约束条件”下生成译文，并在质量闭环触发时进行针对性修正。其核心思想是：把影响长文本翻译稳定性的要素（风格、术语、历史上下文、相似示例）显式注入提示词，让模型在生成时“可被约束”；同时把评估反馈结构化，让修正阶段“可被指导”。

3.4.1 多步引导翻译（Multi-step Prompting）

系统采用多步提示词将翻译过程显式拆解为“理解解构→参考示例→（隐式）多版本思考→融合润色”，并注入：

- **风格元数据**：来自风格识别阶段的领域/语体/复杂度
- **术语表**：全局术语表 + 当前章节术语表（强制遵守）
- **历史上下文**：
    - 相似翻译示例：从已翻译的文本对中召回风格/措辞相近的示例，用于“风格模仿”与“术语用法对齐”
    - 跨章节记忆：提供前序章节的稳定表达方式，抑制风格漂移
    - 本章上下文：提供本章此前已译内容，提升指代一致与语篇连贯

> 注：系统并不要求模型输出多版本，而是将“多版本生成”作为隐式推理过程，以降低输出噪声并保持最终产物单一。

3.4.2 针对性修正（Targeted Refinement）

当评估阶段判定未达标时，系统把最近一次评估信息（错误类型、具体问题、改进建议、回译文本）注入修正提示词，形成“只改错、不重写”的约束式修正策略，强调：

- 保留正确部分，避免过度改写。
- 严格遵守术语表。
- 结合回译文定位语义漂移。

3.5 Agent 3：评估与自我修正智能体（Evaluation Agent, TEaR）

该智能体负责把“质量”转化为可计算、可记录、可反馈的结构化信号。系统采用 TEaR 思路：先通过回译产生“可对比的证据”，再在多维度上生成评分与可操作的修正建议，并将其写入证据链用于后续迭代与审计。

3.5.1 回译验证（Back-Translation）

系统将 `combined_translation` 回译为英文 `back_translation`，并要求：

- LaTeX 公式保持不变（避免公式污染评估）。

3.5.2 结构化质量评估（Structured Quality Review）

系统用 `QualityReview` 结构输出评估结果：

- `score`（0-10）与 `pass_flag`
- `critique`（总体意见）
- `error_types`（错误类型列表）
- `specific_issues`（可定位问题）

- `improvement_suggestions`（可操作建议）

评估结果写入 `refinement_history` 作为可追溯审计记录，支撑后续分析与报告展示。

3.6 人机协作（Human-in-the-loop）与章节级控制

系统在人机协作上采用**章节级批量审查**（而非逐 chunk 中断），对应 `try/main.py` 的控制流：

- **Phase 1**：自动翻译所有 chunks。
- **Phase 2**：汇总全章术语（去重）并（可选）人工审查。
- **Phase 3**：生成章节摘要（用于后续章节上下文）。
- **Phase 4**：章节级质量审查（支持重译循环）。

章节级策略的动机是：减少中断成本，并在"术语首次出现后"尽快形成可复用的全局约束。

3.7 工程鲁棒性：自动模式的速率限制

在 `--no-human-review` 自动模式下，系统通过 `RateLimiter` 控制 LLM 调用速率，避免触发"每分钟 20 次"限制：

- 滑动窗口统计 + 最小间隔约束
- 在每次 `llm.invoke(...)` 前 `wait_if_needed(...)`

4. 实现细节（Implementation Details）

4.1 基础框架与模块划分

- **工作流**：LangGraph（`StateGraph` + `MemorySaver`）
- **LLM接口**：LangChain（`llm.invoke` + `with_structured_output`）
- **RAG/术语库**：Elasticsearch（`try/rag/es_retriever.py`）
- **持久化**：`output/{book_id}/chapter_{chapter_id}/chunk_{chunk_id:03d}.json`

4.2 输出格式与可解析性

关键节点采用结构化输出：

- 风格识别：`StyleMetadata`
- 术语条目：`TermEntry`
- 质量评估：`QualityReview`

当结构化输出失败时，保留回退机制（默认值/手动解析），确保流程可运行。

4.3 结果文件与追溯字段（Auditability）

每个 chunk 保存：

- `source_text / translation / quality_score`
- `glossary`
- `refinement_history`（每次迭代的评估详情）
- `revision_count`

这使得后续报告可以对：术语一致性、修正次数、错误类型分布等进行统计（本报告结果部分先占位）。

5. 实验（Experiments）【结果占位】

5.1 数据与评测设置（Datasets & Protocol）

实验数据：CVPR/ICCV风格学术论文（AlexNet/VGG/ResNet），按章节切分，顺序翻译，模拟“书籍式连续章节”场景。

评测协议：

- chunk 级别评估（由 `QualityReview` 产出）
- chapter 级别审查（人工/自动）
- 全局术语一致性统计（基于 `reviewed_glossary` 与 chunk 输出）

5.2 指标（Metrics）

本项目的评测指标来自 `try/reports/*_evaluation_metrics_summary.json`，可分为三类：

- **流程内质量分（quality_score）**：系统在 TEaR 评估阶段给出的 1–10 质量分，强调“语义准确、术语一致、风格一致、中文可读”。它能反映系统自洽性，但仍带有 LLM 评估的不稳定性。
- **无监督指标（unsupervised）**：如 `back_translation`、`terminology`、`fluency`、`number_preservation`、`length_ratio` 等，用于从一致性、术语遵循、数字保持、长度合理性等角度刻画质量。
- **有监督相似度指标（supervised）**：如 `bleu`、`semantic_similarity`、`edit_distance`，依赖参考译文（`try/data/*_ch.json`）。这类指标对“同义改写/风格化表达”非常敏感：译文更自然未必意味着 BLEU 更高。

因此，后续对“指标下降”的分析遵循一个基本原则：**先判断是否是评测集合/参考差异导致，再判断是否是系统质量退化导致。**

5.3 Baseline 与对照组（Baselines）

本节主要对比两种运行模式（与 `try/output` 目录对应）：

- **无人工介入（nohuman）**：关闭章节级术语审查与章节级质量审查，术语表直接“自动接受”，并输出到 `output/*_nohuman/`。
- **人工介入（human）**：启用章节级术语审查（可修改术语译名）与章节级质量审查（可提出修改意见并触发重译），输出到 `output/*/`。

这两种设置在工程上最大差异是：**术语库被人工“校准”的程度**不同，以及**重译触发**的可能性不同；二者会直接影响术语一致性与风格稳定性。

5.4 结果汇总（Main Results）

下表汇总了四个数据集（VGG/YOLO/ResNet/U-Net）在“人工介入 vs 不介入”两种模式下的核心指标均值（来自 `*_evaluation_metrics_summary.json`）。其中 `valid_chunks` 表示实际参与评测的 chunk 数。

> **表 1：指标汇总（Human vs NoHuman）**

数据集	设置	valid_chunks	quality_score ↑	back_translation ↑	terminology ↑	bleu ↑	semantic_similarity ↑	edit_distance ↓
VGG	human	19	8.53	5.22	5.62	1.36	7.77	0.73
VGG	nohuman	19	8.74	4.88	5.82	1.34	7.78	0.74
YOLO	human	17	8.24	4.57	5.71	1.21	7.80	0.58
YOLO	nohuman	17	8.35	4.73	5.98	1.30	7.82	0.58
ResNet	human	13	8.23	4.77	6.18	1.89	7.41	0.69
ResNet	nohuman	13	7.69	4.73	6.88	1.90	7.31	0.65
U-Net	human	9	8.78	3.76	6.94	0.87	7.28	0.36
U-Net	nohuman	9	8.33	4.89	6.91	0.88	7.29	0.36

5.4.1 为什么会出现“部分指标下降”?

从表 1 可见：ResNet 与 U-Net 的 `quality_score` 在人工介入后上升，而 VGG/YOLO 的 `quality_score` 均值略有下降；同时存在“某些指标上升、另一些下降”的现象。我们将原因归纳为四类（按优先级从高到低）：

- **（A）评测集合差异导致的不可比性（以 VGG 为典型）**：VGG 的 `evaluation_info.total_chunks` 在 human 为 29，而 nohuman 为 19；虽然二者 `valid_chunks` 都是 19，但这通常意味着两次运行的“输出覆盖范围/章节划分”并不完全相同，因此均值差异不能简单解读为质量退化。
- **（B）“参考译文依赖”的指标对同义改写敏感**：`bleu/semantic_similarity/edit_distance` 依赖参考译文。当人工介入将表达改得更符合中文学术写法（例如把直译修正为更自然的改写）时，可能**提升可读性却降低 BLEU**。
- **（C）RAG 增强带来的“召回-噪声”权衡**：更新后的 RAG 在提高可用证据的同时，也更容易召回风格不同/译法不同的条目；若人工未对关键术语进行足够校准，可能造成 `terminology` 指标波动，进而影响综合分。
- **（D）指标之间的天然 trade-off（以 U-Net 为典型）**：U-Net 的 `quality_score` 上升但 `back_translation` 下降，说明译文可能更符合中文表达（凝练/顺序调整/同义替换），导致回译相似度下降。这属于“更自然 ≠ 更易回译”的典型现象。

5.5 定性对比：RAG检索与人工审查带来的质量提升样例

本节从 `try/output/*` 中选取同一段原文在不同模式下的译文对比。选择原则是：**nohuman_norag**（禁用RAG且无人工审查）在术语/表述上存在明显瑕疵，而 **nohuman**（启用RAG但无人工审查）或 **human**（启用RAG且有人工审查）通过RAG检索或术语审查带来可解释的提升。

样例 1（VGG，人名翻译前后不一致：Krizhevsky 的翻译错误）

- **原文（节选）**：`... inspired by Ciresan et al. (2011); Krizhevsky et al. (2012).` 和 `... (Krizhevsky et al., 2012; Zeiler & Fergus, 2013; ...)`
- **nohuman_norag（无RAG，无评估）**：同一篇文章中，`克里日夫斯基等人（2012）` 和 `Krizhevsky等人，2012年` 混用，**前后翻译不一致**
- **nohuman（有RAG，质量分 8.0）**：统一为 `Krizhevsky等人（2012年）`，**保持一致性**
- **改进点**：**人名翻译前后不一致**是禁用RAG时的典型问题。在同一篇文章中，`Krizhevsky` 应该统一为英文原名（学术惯例）或统一为音译"克里日夫斯基"，不能混用。禁用RAG时，模型缺乏全局术语一致性约束，导致同一人名在不同位置出现不同译法；启用RAG后，通过检索翻译记忆库，能够确保同一术语在全文中保持一致。

样例 2（ResNet，专业术语误译：identity shortcuts 的翻译错误）

- **原文（节选）**：`On this dataset we use identity shortcuts in all cases (i.e., option A).`
- **nohuman_norag（无RAG，无评估）**：`在这个数据集上，我们在所有情况下都使用身份快捷方式（即选项A）。`
- **nohuman（有RAG，质量分 7.0）**：`在这个数据集上我们所有情况下都使用恒等映射连接（即选项A）。`
- **改进点**：**"identity"** 在数学和深度学习中应翻译为"恒等"（identity mapping = 恒等映射），而非"身份"（identity在身份识别语境中的译法）。禁用RAG时，模型可能混淆不同领域的术语含义，将数学中的"identity"误译为身份识别中的"身份"；启用RAG后，通过检索能够获得正确的"恒等"译法。这是**明显的专业术语翻译错误。**

样例 3（YOLO，专有名词误译：You Only Look Once 的翻译错误）

- **原文（节选）**：`We present YOLO, a new approach to object detection.`
- **nohuman_norag（无RAG，无评估）**：`我们提出了YOLO（你只活一次），一种新的目标检测方法。`
- **nohuman（有RAG，质量分 9.0）**：`摘要：我们提出了YOLO，一种新的目标检测方法。`
- **改进点**：**"You Only Look Once"** 的正确含义是"你只看一次"（强调单次检测），而非"你只活一次"（这是对YOLO缩写的字面误译，混淆了"look"和"live"）。禁用RAG时，模型缺乏对专有名词的上下文理解，容易产生**明显的字面误译**；启用RAG后，通过检索能够识别YOLO作为专有名词应保留原样，或提供正确的解释。

样例 4（ResNet，专业术语误译：normalization 的翻译错误，句子顺序有误）

- **原文（节选）**：`We adopt batch normalization (BN) [16] right after each convolution and before activation.`
- **nohuman_norag（无RAG，无评估）**：`我们遵循[16]，在每个卷积后和激活前采用批量归一化（BN）[16]。`（正确）
- **nohuman（有RAG，质量分 7.0）**：`在每个卷积后和激活前我们采用批归一化[16]。`（评估建议改为"批量归一化"）
- **改进点**：**"normalization"** 在深度学习中应翻译为"归一化"（如 Batch Normalization = 批量归一化），而非"规范化"。虽然两种模式下都使用了"归一化"，但禁用RAG时可能出现"规范化"的误译。这是**明显的专业术语翻译错误**，因为"规范化"通常对应"standardization"，而"归一化"对应"normalization"。

样例 5（VGG，专业术语误译：prior art 的字面误译）

- **原文（节选）**：`Our design choices are then discussed and compared to the prior art in Sect. 2.3.`
- **nohuman_norag（无RAG，无评估）**：`我们的设计方案随后在第2.3节中进行讨论，并与先前的艺术作品进行比较。`
- **nohuman（有RAG，质量分 8.0）**：`我们的设计选择随后在第2.3节中讨论，并与先前的工作进行比较。`（评估建议改为"先前的研究"或"先前的技术"）
- **改进点**：**"prior art"** 在学术论文中特指"先前工作/先前研究"，而非字面意义的"艺术作品"（art在这里指"技术/方法"，不是"艺术"）。这是**明显的专业术语字面误译。**禁用RAG时，模型缺乏翻译记忆库的约束，容易产生字面误译；启用RAG后，虽然检索到相关记忆，但若未经过人工校准，仍可能出现术语不一致。这体现了RAG检索在提供"证据"的同时，也需要人工审查来确保术语的学术规

范性。

6. 局限性（Limitations）

- **评估仍依赖LLM**：质量评估与修正建议存在主观性与不稳定性，需要结合人工抽检或外部指标（如 MQM 人工标注）增强可信度。
- **RAG检索为关键词/传统ES**：对"概念等价但词形变化大"的术语可能召回不足，后续可引入向量检索或混合检索。
- **章节级审查延迟**：如果某术语在章节早期出现而后频繁复用，章节末审查会导致中间 chunk 先产生不一致译名（需回写修正）。

7. 复现说明（Reproducibility Checklist）

- **入口脚本**：`try/main.py`
- **工作流定义**：`try/core/graph.py`
- **节点实现**：`try/core/nodes.py`
- **RAG检索**：`try/rag/es_retriever.py`
- **输出目录**：`try/output/`
- **运行模式**：
  - 人工审查：默认模式
  - 自动模式：`--no-human-review`（启用速率限制保护）

附录 A：核心函数与职责映射

- **工作流构建**：见 `try/core/graph.py`
- **节点逻辑实现**：见 `try/core/nodes.py`
- **RAG检索与术语库操作**：见 `try/rag/es_retriever.py`
- **主控与人机协作**：见 `try/main.py`