

=====

ITEM #84 – DBM Method Graph (方法亲疏图) – 设计总览

Conversation Title: Karpathy 访谈分析

Date: 20251102

Authors: Sizhe Tan & GPT-Obot

=====

# ITEM #84 – DBM Method Graph (方法亲疏图) — 设计总览

## 1) 节点与坐标 (Concept Nodes & Coordinates)

- **节点类型** : Algorithm (算法/引擎) 、 Principle (原则/范式) 、 Tooling (工具/索引) 、 Product/UseCase (应用/产物) 、 Meta (方法论/演化) 。
- **多重坐标** (每个节点都有这几类向量/标量)：
  1. **Lexical embedding** : 基于节点名称/别名/关键词的文本嵌入；
  2. **Definition embedding** : 基于“1–3 句标准定义”的嵌入（减少歧义与噪声）；
  3. **Context embedding** : 来自你项目文档/代码注释/使用片段的嵌入（内部语境）；
  4. **Anchor-projection coords (Perspective Distance)** : 投影到一组“锚轴”（见 §2）得到的低维可解释坐标；
  5. **Meta tags** : modality{vision/text/...} 、 search{index/knn/2phase} 、 causal{yes/no} 、 granularity{local/global} 、 online{yes/no} 、 determinism{rule/prob} 等。

实务建议：最终相似度 =  $0.4 * \text{lex} + 0.3 * \text{def} + 0.3 * \text{ctx}$  再与 anchor-projection 的“曼哈顿/马氏距离”做早停筛。

## 2) 锚轴体系 (Anchor Space)

为可解释与对齐，定义 8–12 条“认知主轴”（可后续扩展）：

- **四骏主轴**：Perception / Rules / Integration / Evolution
- **方法轴**：Metric vs Symbolic、Causal vs Correlative、Generative vs Discriminative、Online vs Batch、Local vs Global、Deterministic vs Probabilistic、Discrete vs Continuous、Mono- vs Multi-modal
- **时空尺度轴**：Event-scale / Session-scale / Corpus-scale

每个节点通过与锚点的余弦相似度，获得一组可解释坐标；这既便于分层聚类，也便于“人类审校”。

## 3) 图构造 (Graph Building)

- **边类型**（多关系图）：
  - SIM：相似度 (KNN + 互选 + 阈值  $\tau$ )
  - PREREQ：前置/依赖（由定义/引用/手工标注/代码依赖推断）
  - DERIVES：派生/改进
  - COMPOSES：组合共现（在同一 pipeline/章节/代码中并用）
  - CONTRASTS：对立/替代（如 @unalignedAND vs k-means）
- **结构**：先构建  $\text{SIM KNN}$  图 → 运行 Louvain/Leiden 社区发现 → 社区内再抽取 PREREQ/DERIVES/COMPOSES 强边，形成方法簇与骨干亚树。

## 4) 分层上卷 (Differential Tree over Methods)

- 将方法嵌入空间做差分树（你的拿手）：

- 上层为“范畴/主题章”（如 Metric-Space Family、Causal-Reasoning Family）；
- 中层为“方法簇”（如 2-phase 搜索簇、Blocks-Indexing 簇）；
- 底层为“具体方法节点”。
- 这棵“方法差分树”== DBM 的 Chapter/Section 自动生长器（Chapter 9 之“Tech stack review and reflection”就成为树的一层）。

## 5) APTGOE 的 A：从方法图出发的“自主探索”

让“方法图”驱动自主产出新方法（方法的生成也是产品）：

- 探索策略（可并行）
  - *Coverage-seek*：优先补齐稀疏/盲区锚轴；
  - *Novelty-seek*：最大化与现有组合的“结构新颖度”（图同构差 + 语义差）；
  - *What-If Engine*：在 COMPOSES 边附近，替换/插入节点，虚拟跑通 pipeline；
  - *Minimal-Evolution Threshold*：对每个新组合，评估最小可行提升 ( $>\varepsilon$  即保留)。
- 评价回路 (P,O,E)
  - P：在合成任务集上做 proxy 指标（召回/延迟/稳健）；
  - O：对指标与假设做后验 (posteriors) 更新；
  - E：进化保留（版本化节点/边），失败分支也保留为“负样本锚”。

## 6) 检索与快速决策 (Fast Rule 化)

- 查询→方法：把用户问题/任务向量化为 LHS，在“方法差分树+KNN”上做两步搜索（粗召回→重排序），返回方法包 (methods bundle) 而非单一工具。
- 方法→执行：用 blocksIndexing/2phasesSearch 作快速决策器 (LHS 聚焦 → RHS 方法组合)。

## 7) 数据结构 (Java 轮廓，贴合你的生态)

```

class MethodNode {
    String id, name, type;          // Algorithm/Principle/Tooling/...
    List<String> aliases, tags;     // @perspective, @CCC, ...
    double[] embLex, embDef, embCtx; // 三类嵌入
    double[] anchorCoords;         // 锚轴投影坐标
    Map<String, Object> meta;      // modality/granularity/online/...
}

class MethodEdge {
    String from, to;
    EdgeType type;                // SIM, PREREQ, DERIVES, COMPOSES, CONTRASTS
    double weight;                 // 语义/结构权重
}

class MethodGraph {
    Map<String, MethodNode> nodes;
    List<MethodEdge> edges;
    // 索引
    HnswIndex embIndex;           // KNN 用
    CellsPartition anchorGrid;    // 低维 anchor 空间的格子索引 (快速局部检索)
}

```

存储可选：本地 JSON + HNSW 索引；或 Neo4j（多关系图）；或你现有的 DBM 索引层。

## 8) 构建流水线（可直接落地）

1. 规范化词表：统一 `@tags`、别名、驼峰 → 短定义（1–3 句）。
2. 生成三类嵌入：lex/def/ctx；做向量归一化。
3. 锚轴投影：计算 `anchorCoords`；保存低维坐标。
4. KNN + 互选：建立 `SIM` 边；阈值  $\tau + \text{top-k}$ ；过滤孤立点。
5. 多关系抽边：从文档/代码/清单抽取 `PREREQ/DERIVES/COMPOSES/CONTRASTS`。
6. 社区发现：Louvain/Leiden → 形成方法簇；
7. 差分树上卷：在 anchor 空间分层分割 → 章/节/小节；
8. 评估：
  - 结构指标：模块度 Q、簇内密度/簇间分离；

- 语义指标：簇内方差、锚轴覆盖度；
  - 检索指标：召回@k、决策延迟（ms）、稳定性（多次查询一致性）；
9. **A-探索循环**：Coverage/Novelty/What-If → 指标门控 → 版本化；
10. **Fast Rule 化**：把检索出的方法包映射到执行模板（RHS）。

## 9) 最小原型映射（你列举的标签先落点）

Tag	类型	归属簇（建议）	关键锚轴坐标（示意）
@perspective	Principle	Integration/Anchor 簇	Integration↑, Multimodal↑
@metricdistance	Algorithm	Metric-Space 簇	Metric↑, Deterministic↑
@2phasesSearch	Algorithm	Search/Index 簇	Search↑, Latency↓
@differentialTree	Algorithm	Metric-Space 簇	Hierarchical↑, Local↑
@CCC	Principle	Alignment/Integration 簇	Alignment↑, Multimodal↑
@APTOE	Meta	Evolution/Autonomy 簇	Evolution↑, Autonomy↑
@unalignedAND	Algorithm	Comparative/Decision 簇	Contrast↑, Prob/Deter mix
@whatIfEngine	Tooling	Simulation/Design 簇	Generative↑, Causal↑
@segmenting	Algorithm	EventOps/Time 簇	Temporal↑, Local↑
@generative	Principle	Generative 簇	Generative↑
@thirdeye	Product	Market/Signals 簇	Application↑
@DBM-AI	Meta	Meta/Framework 簇	Integration↑
@eventLanguageModel	Algorithm	EventOps/ELM 簇	Temporal↑, Language↑
@blocksIndexing	Tooling	Search/Index 簇	Index↑, Latency↓
@posteriors	Principle	Bayesian/Uncertainty 簇	Probabilistic↑
@minimalEnvolutionThreshold	Principle	Evolution/Criteria 簇	Evolution↑, Threshold↑
@anchorAndAlignment	Principle	Alignment 簇	Alignment↑
@structuredKnowleges	Principle	Symbolic/Graph 簇	Symbolic↑

Tag	类型	归属簇（建议）	关键锚轴坐标（示意）
@symbolicModels	Algorithm Symbolic/Rules 簇		Rules↑, Deterministic↑

这些先做“种子节点”，随语料增长自动扩展。

## 10) 输出形态（让它“能用起来”）

- **Chapter Builder**：从方法差分树自动生成“章节目录 + 关系图 + 小节摘要”的可读物（你常要的 PDF/海报都能一键生成）。
  - **Method Bundle Recommender**：输入任务描述 → 返回“方法组合 + 执行模板”。
  - **A-Explorer Dashboard**：展示 Coverage/Novelty 探索前沿、方法新枝、失败分支（负样本锚）。
-