GraphAI

by **Receptron team**

GraphData

NodeとEdgeで構成/有向非巡回グラフ

- 非巡回グラフの動作を繰り返すloop
- Nested Graphで繰り返し処理 / JSON, YAML, TypeScriptで記述

Agent

TypeScriptで書かれたプログラム

- llm / RAG / Database / template / http client / echo

GraphData

```
{
  version: 0.5
  graph: {
    llm: {
      agent: "openAIAgent"
      params: {system: "foo bar"}
    },
    template: {
      agent: "stringTemplate",
      inputs: {message: ":llm"}
    }
  }
}
```

Agent

```ts
// Params, result, input
export const dataSumTemplateAgent: AgentFunction<Record<string, any>, number, number> = async ({ inputs }) => {
  return inputs.reduce((tmp, input) => {
    return tmp + input;
  }, 0);
};

const dataSumTemplateAgentInfo: AgentFunctionInfo = {
  name: "dataSumTemplateAgent",
  agent: dataSumTemplateAgent,
  samples: [
    {
      inputs: [1],
      params: {},
      result: 1,
    },
    {
      inputs: [1, 2],
      params: {},
      result: 3,
    },
  ],
  description: "Returns the sum of input values",
  category: ["data"],
  author: "Satoshi Nakajima",
  repository: "https://github.com/receptron/graphai",
  license: "MIT",
};
export default dataSumTemplateAgentInfo;
```

npm

- graphai 本体
- @graphai/*_agents - agent
  - 単機能のごとに１つのnpm=agent / 依存関係を減らす目的
  - @graphai/vanilla - npmの依存のないagent
  - @graphai/llm_agents - openAIAgent, groqAgentなどのメタパッケージ
  - @graphai/agents - 全部入り
- @receptron/* ツール郡
  - graphai_cli, graphai_express, agent_filters

動作方法(図をいれて１つ１つ丁寧に説明)

- クライアント(ブラウザ）のみで動く
- サーバのみで動く
  - クライアントからGraphDataをpost
  - サーバにGraphDataこみで処理を実装
- サーバとクライアント連携して動く
  - GraphDataはクライアントで実行
    - サーバで動かす必要のある処理だけサーバで動かす
      - API keyの秘匿性 / データベースへのアクセス / 書き込み
    - AgentがhttpのendpointとAPI対応

AgentFilter

- 各Agentを実行する前後に処理を挟む
  - expressのmiddleware, railsのaround filter
  - agentId, nodeIdで制御
- 例
  - サーバへ処理をバイパス
  - キャッシュ
  - ログ

Streaming

- AgentFilterとAgent側の実装
- httpのstreamingに対応可能
- いずれの動作方法でも可能
- 並列で動いている場合も対応

サーバ/クライアントダイナミック

AgentFunctionInfo

- agentの本体と、agentに関する情報
- GraphAIの動作のみならず、様々なツールで利用可能

ユーティリティ

- Agentテスト
  - AgentFunctionInfoを使ってUnit Test
    - TDD
  - Agentのdoc
    - documentの自動生成
  - express serverのmiddleware
    - すぐにサーバ、クライアント構成

Express Server(API)

- AgentFunctionInfoを元にApiの情報
- 将来的には

--

Future