UNIVERSIDAD NACIONAL DE ROSARIO

TECNICATURA UNIVERSITARIA EN INTELIGENCIA ARTIFICIAL

Procesamiento del Lenguaje Natural (IA4.2)

TRABAJO PRÁCTICO 2

DOCENTES:

- D'Alessandro, Ariel
- Geary, Alan
- Leon Cavallo, Andrea
- Manson, Juan Pablo

ALUMNO:

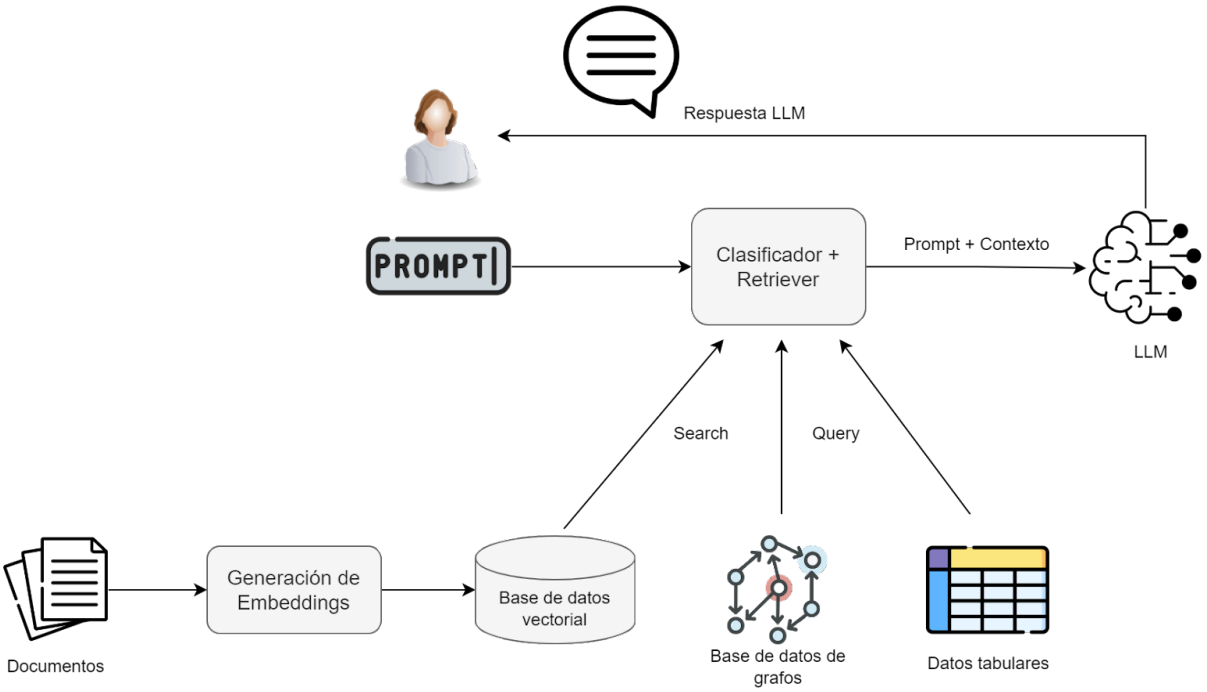
Wagner, Juan (w-0557/6)

AÑO: 2022

Ejercicio 1 - RAG

Crear un chatbot experto en un tema a elección, usando la técnica RAG (Retrieval Augmented Generation). Como fuentes de conocimiento se utilizarán al menos las siguientes fuentes:

Documentos de texto
Datos numéricos en formato tabular (por ej., Dataframes, CSV, sqlite, etc.)
Base de datos de grafos (Online o local)



El sistema debe poder llevar a cabo una conversación en lenguaje español. El usuario podrá hacer preguntas, que el chatbot intentará responder a partir de datos de algunas de sus fuentes. El asistente debe poder clasificar las preguntas, para saber qué fuentes de datos utilizar como contexto para generar una respuesta.

Requerimientos generales

Realizar todo el proyecto en un entorno Google Colab
El conjunto de datos debe tener al menos 100 páginas de texto y un mínimo de 3 documentos.
Realizar split de textos usando Langchain (RecursiveTextSearch, u otros métodos disponibles). Limpiar el texto según sea conveniente.
Realizar los embeddings que permitan vectorizar el texto y almacenarlo en una base de datos ChromaDB
Los modelos de embeddings y LLM para generación de texto son a elección
Para el desarrollo del “Clasificador” es posible utilizar diversas técnicas aprendidas en la materia, por ejemplo en Unidad 3 y Unidad 6

—
La idea original para este RAG era la de poder consultar en forma extensiva e inteligente toda la normativa argentina.
La misma puede encontrarse en el portal https://www.argentina.gob.ar/normativa.

Para realizarlo se tubo que scrapear toda la página en búsqueda de todas sus normas existentes (nacionales y provinciales), tomar de cada resultado encontrado su meta, como el año en que fue sancionada, en que boletin oficial fue publicada, a que otra norma modifica o es modificada, resumen y texto de la misma, entre otras cosas.

Un primer scrapeo identificó todas las normativas existentes desde 1856 a la actualidad.

Luego se prosiguió a realizar en forma conjunta:
- la extracción del texto de las normas
- el armado de un grafo que las pueda ir conectado en función de quien ha sido modificado por quien.

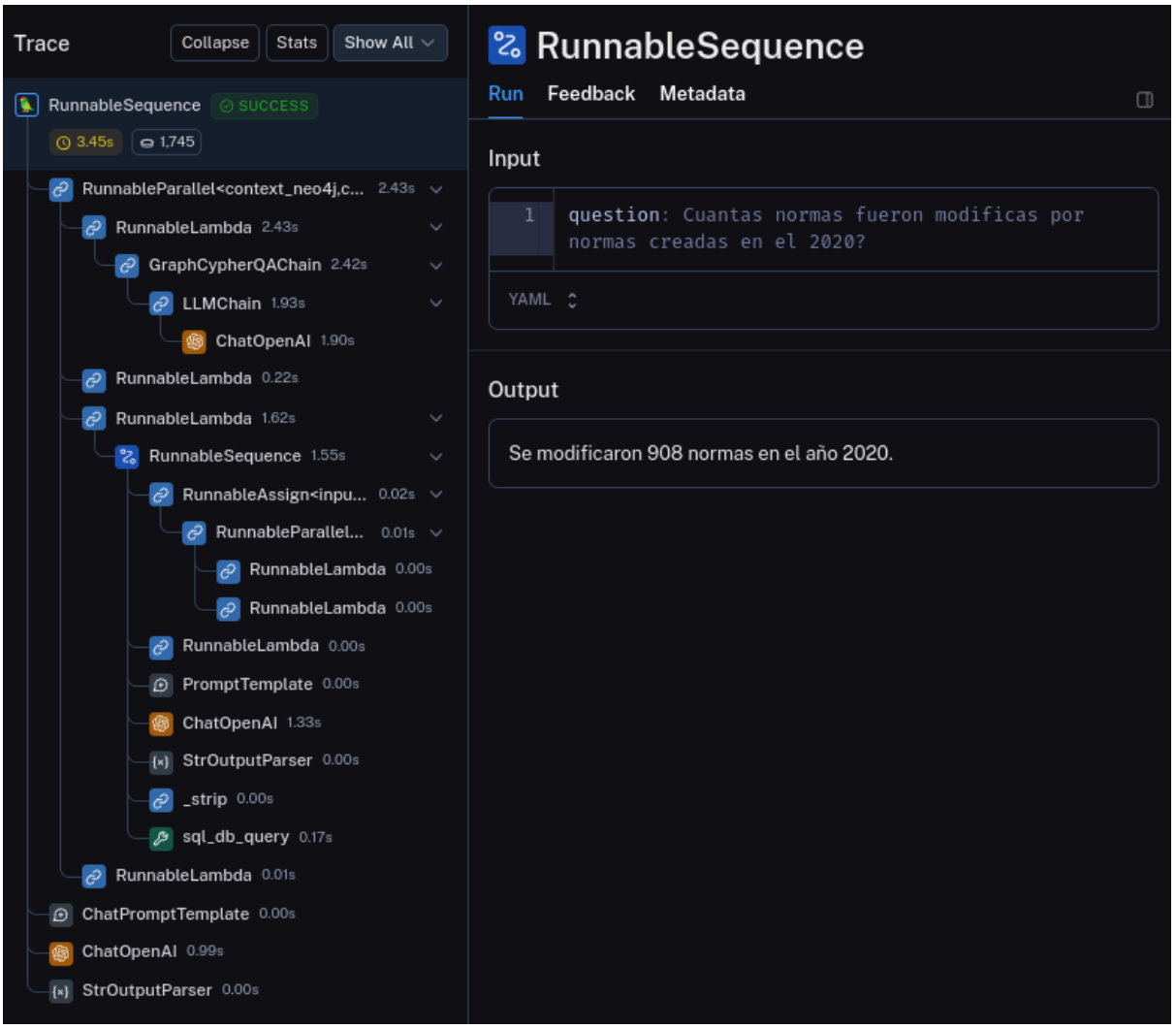
De base relacional se optó por usar sqlite por su extrema sencillez e integración out-of-the-box en Python. A su vez para formar el grafo de conocimiento se utilizó Neo4j dado que ofrece un free tier que luego de analizarlo, NO alcanza para trasladar toda la información normativa.
La versión gratuita ofrece hasta 200k nodos y 400k relaciones. (A la actualidad hay ~425k normas entre nacionales y provinciales).

A medida que se habían extraído todos los textos de las normativas, se pasó a realizar una incrustación de párrafos para insertarlas en una base de datos vectorial (ChromaDB).

Todo fue realizado usando langhain como orquestador/interfaz principal y debugeado con langsmith.

El resultado final es un pequeño chat (que no posee memoria)

Aqui hay una vista el debug provisto por langsmith



Ejercicio 2 - Agentes

Realice una investigación respecto al estado del arte de las aplicaciones actuales de **agentes inteligentes** usando modelos LLM libres.

Plantee una problemática a solucionar con un sistema multiagente. Defina cada uno de los agentes involucrados en la tarea. Es importante destacar con ejemplos de conversación, la interacción entre los agentes.

Realice un informe con los resultados de la investigación y con el esquema del sistema multiagente, no olvide incluir fuentes de información.

Opcional: Resolución con código de dicho escenario.

—

Para la investigacion sobre el estado del arte de las aplicaciones actuales me he basado exclusivamente en el contenido del siguiente repositorio:
https://github.com/hyp1231/awesome-llm-powered-agent

El mismo es un destilado de las diferentes aplicaciones existentes junto a los papers que avalan las pruebas de conceptos realizadas.

Dado que los LLM tiene la capacidad para planificar, razonar, elegir herramientas que les aumenten su conocimiento disponible como así de poder ejecutar acciones a través de API, se vuelven en entidades con casi un sin fin de aplicaciones y de relaciones con el usuario.

La autonomía de estos sistemas basados en LLM es tal que pueden actuar con supervisión humana, en cooperativamente con humanos, o completamente autónomos.

ALGUNOS CASOS DE USO:

I) Autonomous Task Solver
Esta área se centra en el desarrollo de sistemas capaces de resolver tareas de manera autónoma, empleando capacidades de razonamiento general, planificación y uso de herramientas. La innovación en este campo apunta a la creación de asistentes virtuales y agentes de inteligencia artificial (IA) que pueden interactuar con el mundo físico y digital de manera más efectiva, adaptándose a nuevas tareas y contextos sin intervención humana directa.

I.I) General Reasoning & Planning & Tool Using
Aborda cómo los modelos de lenguaje pueden automatizar la solución de tareas mediante razonamiento general, planificación y uso de herramientas. Aquí se hace hincapié en desarrollar agentes capaces de entender contextos complejos, tomar decisiones informadas

y realizar acciones con una mínima intervención humana, adaptándose a diversos entornos y desafíos.
Uno de sus papers más destacados es "ReAct: Synergizing Reasoning and Acting in Language Models" de Shunyu Yao et al el cual se destaca por integrar de forma efectiva el razonamiento y la acción dentro de modelos de lenguaje, permitiendo una mayor autonomía y capacidad de adaptación de los agentes en la resolución de tareas. El aporte fundamental de este paper es como combina técnicas de planificación y actuación para mejorar la eficacia de los agentes autónomos.

I.II) Multi-Agent Cooperation
Explora cómo múltiples agentes autónomos pueden colaborar y coordinarse para alcanzar objetivos comunes. Se centra en el desarrollo de métodos para mejorar la interacción entre agentes, permitiendo una cooperación eficiente dado que cada agente puede especializarse permitiendo resolver tareas complejas y dinámicas.
En el paper "MetaGPT: Meta Programming for Multi-Agent Collaborative Framework" utilizan técnicas de metaprogramación, ej, aprender cómo aprender, reflexión e introspección, entre otros, para facilitar la colaboración entre múltiples agentes, potenciando su eficiencia y su efectividad para la cooperación.

I.III) Framework & Open-Source
Integraciones a software para poder acelerar la innovación y facilitar el acceso a tecnologías avanzadas mediante agentes autónomos para investigadores y desarrolladores, sugiriendo papers relacionados, dando datos fácticos o dando ayuda en términos de programación.

I.IV) Desarrollo Web
Pudiendo efectuar tareas de scrapeo de contenido web sistematizado

I.V) RL Agents
Todo lo concerniente a aprendizaje por refuerzo donde se busca interactuar de manera efectiva entre el mundo físico y virtual.

I.VI) Robótica y Embebidos IA
Busca crear agentes que puedan entender y actuar en el mundo físico a través de la percepción multimodal y el procesamiento del lenguaje natural. Aquí se irrumpe desde la navegación y la planificación de tareas hasta la interacción social y la toma de decisiones autónoma, utilizando el conocimiento textual y la comprensión del lenguaje para mejorar la capacidad de los robots de operar de manera más natural y efectiva en entornos complejos y dinámicos.

I.VII) Gaming & Role-Playing
Busca desarrollar agentes que pueden interactuar, razonar, y tomar decisiones en entornos de juego complejos. La innovación en esta área abarca desde la mejora de la interacción y la narrativa en los juegos hasta la implementación de sistemas tutoriales conversacionales y el análisis de la toma de decisiones estratégicas en juegos de información imperfecta.

I.VIII) Trustworthy
Se enfoca en abordar los desafíos éticos, de seguridad y de confiabilidad asociados con la implementación de Modelos de Lenguaje Grande (LLM). Esto incluye investigaciones sobre

cómo garantizar que los agentes de AI actúen de manera segura, ética, y predecible, así como el desarrollo de métodos para identificar y mitigar riesgos potenciales que puedan surgir de su uso.

II) Human Interaction Simulation
Esta categoría se centra en el uso de LLM para simular interacciones humanas, comportamientos competitivos, dinámicas de redes sociales, y planificación estratégica. Los estudios en esta área buscan entender cómo los agentes basados en LLM pueden imitar personalidades humanas, tomar decisiones estratégicas, y participar en entornos sociales de manera realista.

III) Human-Agent Interaction
Aborda cómo la interacción entre humanos y agentes basados en (LLM) afecta y mejora diversas facetas de la experiencia humana, desde la creatividad hasta la formación de opiniones colectivas y el soporte para el bienestar mental. La investigación aquí busca entender y optimizar la manera en que los agentes de IA pueden colaborar con los humanos, influir en la dinámica social, y proporcionar asistencia emocional y cognitiva.

IV) Agents-Powered LLMs
Se enfoca en el apalancamiento de agentes autónomos a través de LLM especializados, explorando cómo los agentes pueden instruir, mejorar, o interactuar con LLMs para realizar tareas de razonamiento, síntesis de programas, entre otros. Esta área de investigación busca ampliar las capacidades de los LLMs al integrarlos con técnicas de aprendizaje profundo, aprendizaje por refuerzo, y simulaciones de sociedades, con el objetivo de crear sistemas de inteligencia artificial más avanzados y versátiles.

V) Benchmark
Consiste en el desarrollo y la evaluación de benchmarks específicos para LLM que actúan como agentes autónomos o asistentes. Estos benchmarks están diseñados para medir la capacidad de los LLMs para realizar tareas complejas, utilizar herramientas, adaptarse a nuevos entornos, y colaborar con otros agentes. La creación de benchmarks rigurosos es crucial para poder comprar el rendimientos de los diferentes agentes y LLMs.

VI) Survey & Tutorial
Esta categoría se centra en proporcionar una visión general y educativa sobre el estado actual y las posibilidades futuras de los LLM aplicados como agentes autónomos en diversos campos. Los trabajos aquí incluyen tutoriales que explican técnicas de modelado y razonamiento basadas en lenguaje natural, así como encuestas exhaustivas que revisan el progreso, los desafíos, y las direcciones futuras de los agentes basados en LLM. Estos recursos son esenciales para investigadores, desarrolladores y entusiastas de la IA que buscan comprender el impacto transformador de los LLM en la creación de agentes inteligentes.

Estas son las aplicaciones encontradas en dicho repositorio. Para más información, el repositorio cuenta con un listado de los papers relacionado a la temática, junto con una indicación a los mas citados en el ámbito académico y a aquellos que recibieron mayor aceptación por la comunidad.

Con respecto a la problemática a resolver utilizando multiagentes, he elegido un aspecto lúdico.

Quiero convertir la experiencia de jugar juegos de rol, para delegar toda la función del Dungeon Master (DM) -quien decide el rumbo y los eventos que ocurren-, además todo Personaje No Jugador (NPC) tambien es controlado por LLM.

Aquí ya reconocemos que necesitamos algunos agentes y componentes:

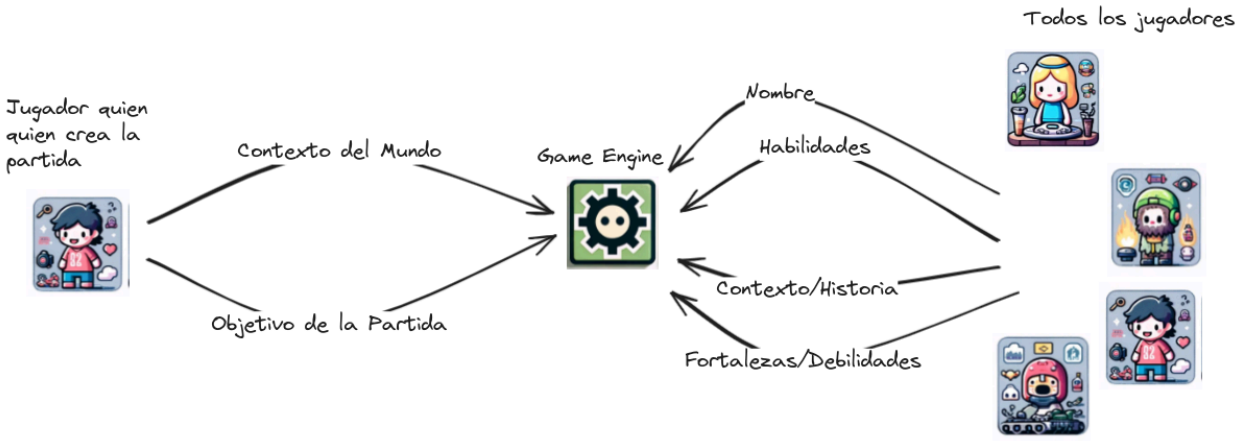
- Agentes:
    - DM
    - NPC
    - Experto en creación de partidas DM

- Componentes:
    - Game Engine: Orquestador de Agentes

Primero explicaré toda la dinámica del juego y luego descomponer los elementos involucrados con una ejemplificación de cómo sería su comunicación.

El juego comienza con un jugador creando una partida. Para crear una partida se debe chatear con el Game Engine para establecer unos lineamientos generales de las características de la partida que que se quiere tener, esto incluye dar contexto o describir el mundo donde se encontrará establecido como así también aclarar el objetivo principal que deben alcanzar los jugadores, ya sea eliminar a un determinado jugador/NPC, capturar una bandera, escapar, liberar a un jugador/NPC, etc.

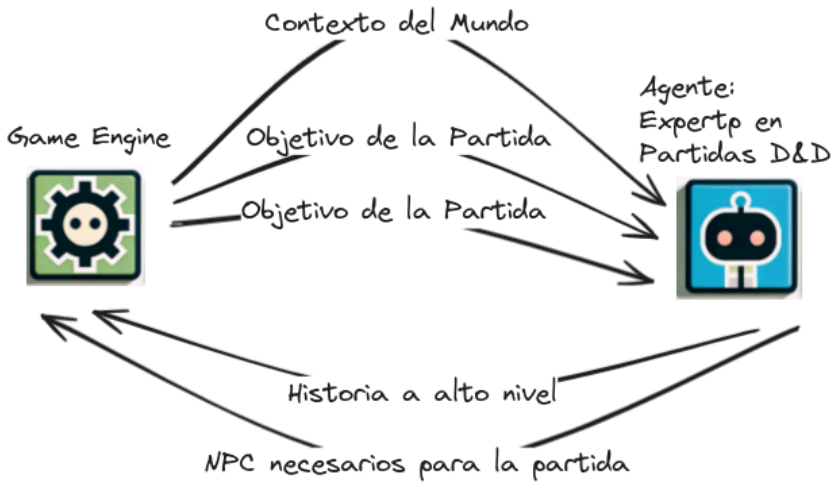
Luego, cada jugador deberá crear su personaje indicando su nombre, dar su propio contexto/historia, decir que habilidades y debilidades tiene, como así también que items cuenta.



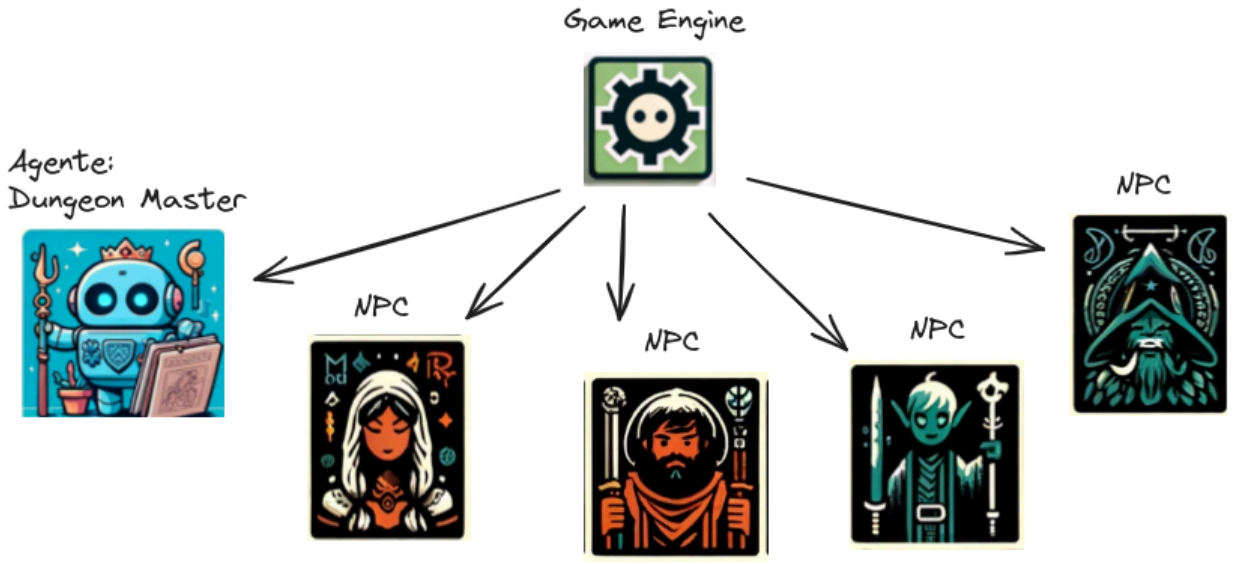
Una vez creados todos los personajes dicho esto, el Game Engine, interactúa con el agente creador de partidas al cual se le pasan los requerimientos de partida junto a todos los personajes intervinientes.

Con toda esa información el agente ahora puede planificar y razonar sobre la partida a crear.
Deberá ser responsable de:

- Determinar una aventura que resulte interesante y divertida, a muy alto nivel.
- Determinar los NPC necesarios para llevar adelante la partida. Estos NPC tienen los mismo atributos que los de los jugadores (Nombre, contexto, historia, fortalezas/debilidades) y además una personalidad y un objetivo propio.



El Game Engine deberá instanciar entonces todos los agentes NPC intervinientes para la ejecución de la partida incluyendo al Dungeon Master, (esta instanciación consiste en definir el contenido de `system` de cada Agente).



El Game engine le solicita al Agente DM para que comience la partida.

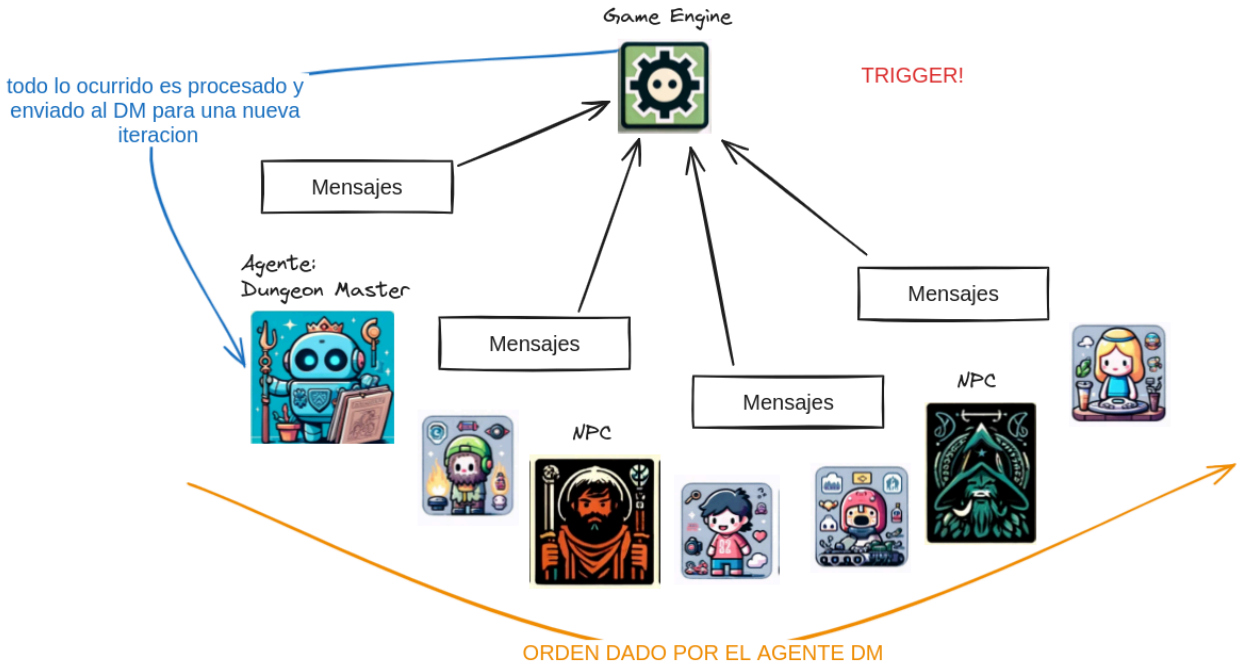
El Agente DM, planificara la primera iteración de juego, la cual da inicio al mismo. Aquí hace una presentación y permite que los jugadores participen.

Los jugadores podrán ir escribiendo texto, participando de forma tan libre que como deseen hasta que algún evento desencadenador, enviara todo lo ocurrido al Dungeon Master para que este vuelva a planificar una próxima iteración.

Estos eventos desencadenadores son dos:

1. Peligra la cantidad de contenido a enviarle al Dungeon Master, es decir, entre toda la información que tiene que tener de entrada y la respuesta del mismo peligra su ventana de contexto.
2. Se ha dicho alguna palabra clave
3. Ha pasado un cierto tiempo
4. Ha habido una cierta cantidad de mensajes

Toda esta información generada viaja al Game Engine para que este pueda ordenarla a modo de ser pasada luego al Dungeon Master.



Se repite el ciclo hasta que o bien el DM haya decido terminar la partida o bien los jugadores hayan conseguido el objetivo primario.

Detalles tecnicos:

Control de Agentes por parte del Game Engine:
Suponiendo el siguiente template:

```
<|im_start|>system
{{ .System }}<|im_end|>
```

```
<|im_start|>user
Debes tomar una acción dado tu prompt basado en tu siguiente contexto.
Contexto:
{{ .Contexto }}
Prompt:
{{ .Prompt }}<|im_end|>
<|im_start|>assistant
```

donde cada `.System` fue establecido por el Agente Experto en Partidas de D&D, el `.Contexto` es un resumen anterior más los últimos mensajes y `.Prompt` es la indicación a actuar que le da el Agente Dungeon Master.

En cada iteración lo único que queda constante es la configuración `.System` de cada NPC él puede puede ser algunos ejemplos:

- Para Agente Dungeon Master: “Este una chatbot comportandote como Dungeon Master que debe seguir esta historia XXXX y permitir tener iteraciones agradables.”
- Para un NPC como un Enemigo: “Eres un chatbot el cual tiene que personificar a una criatura…”

Para el caso del Agente Dungeon Master, su contexto incluirá, información sobre los participantes, es decir, el conoce sus motivaciones, debilidades y fortalezas, y además los mensajes.

Es importante destacar que la suma total de tokens existente en el template, siempre debe ser menor a la ventana de contexto del modelo de lenguaje que luego procese la consulta. Es importante considerar que se debe dejar una cantidad de token para salida también, por eso esto es otra “alarma” para el Game Engine para cortar una iteración.

La contribución de cada agente es poder separar responsabilidades y permitir que se comporten como jugadores reales.