UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

DIVISIÓN DE ESTUDIOS DE POSGRADO

BUSCADOR PERSONALIZADO PARA LA PRÁCTICA AUDITIVA DEL INGLÉS MEDIANTE UN AGENTE NEURONAL

TESIS

PARA OBTENER EL GRADO DE

MAESTRO EN MEDIOS INTERACTIVOS

Presenta:

Ing. Cesar Felipe Martínez Cisneros

Director:

Dr. Ignacio Arroyo Fernández

Huajuapan de León, Oaxaca, Diciembre de 2023

Dedicatoria

Dedico este trabajo de tesis a mi hija, Irina Elisa, quien ha traído paz, amor y esperanza a esta nueva etapa de nuestras vidas.

Agradecimientos

Agradezco a Dios por las bendiciones recibidas, tanto para mí como para mi familia y amigos.

A mi esposa, Adriana, le agradezco el tiempo que me permitió dedicar a esta maestría. Su amor, esfuerzo, apoyo y comprensión fueron fundamentales en este proyecto. Agradezco a mi hija, Irina, por las mañanas y noches en las que no me veías salir ni llegar a casa. Hoy, juntos, lo hemos logrado.

Agradezco a mi mamá, Fidelia, y a mi papá, Francisco, por ser siempre un puerto seguro al que puedo recurrir para llenarme de energía, amor y sabiduría. Gracias por todas sus oraciones, apoyo y comprensión.

Expreso mi agradecimiento a mi suegra, Asunción, y mi suegro, Mario, por todas las atenciones y hospitalidad que me han brindado en su casa.

De igual forma, agradezco de todo corazón a mis hermanos, Alfredo, Rafael y Karina, por el apoyo que siempre me han brindado en cada etapa de mi vida. Siempre los llevo presentes en mi corazón.

Reconozco y aprecio la dirección del Dr. Ignacio Arroyo Fernández, quien dedicó su tiempo y conocimiento para que este proyecto terminara exitosamente. Es importante destacar que el Dr. Ignacio fue mi compañero de carrera y hoy tengo el honor de ser su tesista de maestría. Gracias, amigo Nacho.

Mi reconocimiento y gratitud para los sinodales de tesis, Dr. Anibal Arias, Dr. Eduardo Soto y Dra. María Pinto, quienes supervisaron y evaluaron el desarrollo de este trabajo.

Gracias a mi amigo y compañero de clases, Owen, por compartir el aula estos dos años. A mi amigo Gerardo y su hermano Francisco por apoyarme en los proyectos audiovisuales. Al profesor Carlos por permitirme aprender más sobre el usalab.

Agradezco al Dr. Mario Moreno Rocha por todos los conocimientos en el campo de la interacción humano-computadora. Así también, a todo el núcleo de profesores de la maestría en medios interactivos por su esfuerzo y apoyo.

Mi sincero agradecimiento a la Universidad Tecnológica de la Mixteca, a mi jefe inmediato, el profesor Jonathan Dolan, y al vice-rector administrativo, Javier José Ruiz Santiago, por brindarme su apoyo durante mis estudios de maestría.

Resumen

Actualmente, los buscadores de información dentro de la informática se han convertido en herramientas indispensables para los usuarios. Estas tecnologías han brindado un apoyo significativo en el ámbito educativo, especialmente en el aprendizaje de una segunda lengua, como el inglés. Los buscadores han facilitado el acceso a materiales de práctica para el desarrollo de las cuatro habilidades lingüísticas, siendo la habilidad auditiva una de las más difíciles de adquirir entre los estudiantes. En ese contexto, se han detectado limitaciones en estas herramientas en cuanto a su adaptabilidad a las necesidades individuales del estudiante. En específico su nivel de dominio de la lengua y su desconocimiento de vocabulario presente en las conversaciones de práctica en la base de datos. Por lo anterior, el presente trabajo de tesis desarrolla una herramienta denominada 'Buscador Personalizado para la práctica auditiva del inglés mediante un agente neuronal'. La tarea principal del agente es aprender a realizar consultas más efectivas al buscador mediante la técnica de aprendizaje por refuerzo. Para lograr el aprendizaje, el agente interactúa con un entorno que consiste en una base de datos y un buscador convencional. Las consultas generadas por agente constituyen sus acciones, mismas que ingresan al entorno y alteran su estado. Las salidas de este último incluyen una recompensa para el agente y los estados actuales del entorno. Si la consulta generada por agente es similar a los requisitos del usuario, la recompensa es positiva; de lo con-

trario, es negativa. Los resultados mostraron que a mayores recompensas en promedio, se obtienen conversaciones de práctica más adaptadas a las necesidades usuario.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La creciente integración de las herramientas tecnológicas en el campo educativo ha implicado un sustancial apoyo a los estudiantes en el aprendizaje de una segunda lengua [1]. El caso particular de la lengua inglesa no es la excepción, ya que los docentes se apoyan de diversas herramientas digitales para la enseñanza de sus cuatro habilidades fundamentales: escrita, lectora, oral y auditiva, siendo la habilidad auditiva la que presenta mayor dificultad para dominar entre los estudiantes [2]. Una perspectiva altamente promisoria radica en la utilización de tecnologías emergentes, como los sistemas basados en Inteligencia Artificial. Estas tecnologías ofrecen un sustancial beneficio al proceso educativo cuando se ponen al alcance de docentes y estudiantes.

Diversas herramientas tecnológicas han sido utilizadas para mejorar la escritura, la pronunciación, el aprendizaje de vocabulario; entre otras habilidades. No obstante, para la comprensión auditiva de manera específica es difícil encontrar una herramienta basada en los requerimientos del usuario en cuanto a su nivel de dominio, mismo que los estudiantes buscan mejorar escuchando conversaciones grabadas. En este sentido, los requerimientos personalizados de los usuarios también involucran su conocimiento previo

sobre el vocabulario disponible en los repositorios digitales de ejercicios.

Este conocimiento previo con frecuencia es poco al inicio, lo que dificulta la búsqueda de los recursos y la obtención de resultados más satisfactorios para el usuario. De igual manera, el usuario tiene diferentes requerimientos sobre la complejidad de los ejercicios auditivos que necesita realizar, ya que en el repositorio de recursos auditivos se pueden encontrar conversaciones con vocabularios más complejos que otros. Esto hace a unas conversaciones más complejas que a otras.

Por esta razón, la presente investigación pretende crear un método de Inteligencia Artificial (IA), en particular basado en Aprendizaje por Refuerzo (RL, por sus siglas en inglés), que permita realizar búsquedas de conversaciones de manera personalizada como apoyo para la práctica de la habilidad de comprensión auditiva en la lengua inglesa. Se enfatiza que "personalizada" se refiere a que el usuario puede ingresar tanto un tema de su interés como un grado de complejidad de vocabulario que requiera, de manera que el método los adapta cada vez mejor a los resultados de búsqueda, de acuerdo con el nivel de dominio del usuario.

Es importante destacar que existen investigaciones previas similares a la propuesta planteada en este trabajo. Un ejemplo de ello se encuentra en el estudio presentado en [3], que aborda un generador de consultas en lenguaje natural (SNLQ) a través de un algoritmo de aprendizaje por refuerzo. La generación de consultas es un problema desafiante, por lo que actualmente es una línea de investigación muy activa que intenta mejorar la precisión de las mismas. Además, se puede mencionar otro trabajo previo presentado en [4], titulado 'Sistema de Recomendación de Lecciones para Estudiantes de la Lengua Inglesa'. Este sistema incorpora métodos de Inteligencia Artificial que identifican lecciones con temas similares y las asocian progresivamente a cada usuario. La efectividad de este sistema de recomendación ha sido

demostrada en el contexto del aprendizaje del inglés. El trabajo mostrado en [5] también presenta un sistema de recomendación personalizado de recursos de audio y vídeo, también basado en aprendizaje automático. En todos los casos que se lograron observar hasta ahora, la personalización consiste en asociar temáticas a cada cuenta de usuario. Éste es un enfoque típico en los sistemas de recomendación que en esta tesis se extenderá un poco más hacia los requerimientos del usuario según su nivel de dominio de la habilidad auditiva.

Las antes mencionadas son algunas herramientas tecnológicas encontradas hasta el momento para asistir a los usuarios que quieren mejorar su aprendizaje del inglés. Hay otros sistemas ampliamente utilizados para recomendar material didáctico. Sin embargo, hasta el momento no se ha encontrado un sistema que esté orientado únicamente a la habilidad de comprensión auditiva, como es el caso del método que se desarrolló en este trabajo de tesis. Cabe resaltar que además de ser más difícil de adquirir [2], esta habilidad es también muy importante al aprender una lengua extranjera debido a que representa hasta un 50 % del proceso de comunicación [6]. Durante el aprendizaje de esta habilidad, la audición presenta cambios incrementales de complejidad, ya que en los niveles más altos, B1 y B2 de acuerdo con el Marco Común Europeo de Referencia, los ejercicios de conversación son más parecidos a la vida cotidiana. Esto significa comunicación para la supervivencia en un país de habla anglosajona [7].

Los experimentos realizados usando el método propuesto en esta tesis tienen el potencial de servir de guía para la creación de una interfaz prototipo. Esta solución ligará al usuario con el método propuesto tomando tanto su tema de interés como un nivel de complejidad de vocabulario que desee. Como este método se basa en RL, los parámetros de complejidad de vocabulario y tema de usuario son atributos controlados de las conversacio-

nes recuperadas desde una base de datos o repositorio. Estas conversaciones fueron obtenidas a partir de diversas fuentes académicas de acceso libre. La base de datos está conectada a un buscador convencional conformado por un método de recuperación de información. Este sistema recibe consultas generadas por el agente de RL/control como acciones, las procesa y recupera conversaciones transcritas, las cuales se regresan al agente para que las analice junto con una señal de retroalimentación/"reward". Esta señal le indica al agente si la consulta que emitió es acorde con los requerimientos del usuario o no, lo que le indica, respectivamente, si reforzar o cambiar sus parámetros internos (constituidos por una red neuronal).

Los experimentos realizados tuvieron como finalidad observar si el agente neuronal pudo adaptar las consultas que realiza el usuario al contenido de la base de datos para que las conversaciones recuperadas tengan la relevancia y complejidad adecuadas. Los resultados obtenidos mostraron que hubo factores tanto del entorno que representa la base de datos como del agente que fueron importantes para el rendimiento. Tales factores incluyen el número de palabras de las consultas generadas por el agente, el de las generadas por el usuario, así como el modelado del entorno que implementa la base de datos y la recuperación de información como un proceso secuencial. Estos fueron fundamentales para afectar positiva o negativamente el desempeño del agente al realizar la adaptación cuando se le pidieron consultas y complejidades simuladas de usuario. Tanto la relevancia de los resultados recuperados por el agente como la complejidad de vocabulario de las conversaciones transcritas con respecto a estos requerimientos simulados se incrementaron significativamente después de probar múltiples configuraciones. Además, y no menos importante, se observó un decremento significativo de errores gramaticales en las consultas generadas por el agente mientras se adaptaba a los requerimientos simulados de usuario.

1.1. Planteamiento del problema

Los estudiantes de lenguas extranjeras continuamente realizan búsquedas de material para la práctica de la habilidad auditiva. Muchas de estas búsquedas difícilmente toman en consideración el nivel de dominio alcanzado por el estudiante. Asimismo, el usuario difícilmente conoce el contenido de la base de datos para realizar una búsqueda con los términos correctos para encontrar resultados más precisos.

En este trabajo se propone la construcción de un buscador personalizado de conversaciones transcritas mediante el uso de un agente neuronal a partir de una base de datos. Este agente tendrá el objetivo de adaptar las búsquedas que realice con referencia a un tema de interés y un nivel de complejidad de vocabulario que desee el usuario (requerimientos de usuario). Para tal fin un algoritmo de aprendizaje por refuerzo permite al agente reformular consultas para la base de datos mediante un método de recuperación de información. Este método se implementa como un entorno de aprendizaje a través del cual el agente navega. Este entorno evalúa las transcripciones recuperadas por el agente, comparándolas con los requerimientos del usuario. Si las conversaciones recuperadas cumplen aproximadamente con dichos requerimientos, el entorno devolverá una retroalimentación positiva (recompensa o "*reward*") al agente. El propósito del algoritmo es que el agente obtenga el mayor número de recompensas acumuladas por episodio de entrenamiento. En este caso, se dice que el agente está aprendiendo a realizar mejores consultas a la base de datos, lo que se traduce en una mejor adaptación de las temáticas y complejidades de vocabulario requeridos por el usuario al contenido de conversaciones transcritas de la base de datos. De este modo, las conversaciones transcritas recuperadas pueden ser fácilmente asociadas con sus audios correspondientes en la base de datos para

que el usuario las utilice.

Además de las mejoras en la recompensa acumulada que el agente obtiene por sus consultas generadas, estas también pueden evaluarse en términos de su 'calidad lingüística'. Cuando hablamos de 'calidad lingüística', nos referimos al número de errores gramaticales que el agente comete al formular consultas en el buscador y a la disminución de estos errores a medida que aprende a realizar búsquedas más efectivas.

Los buscadores y sistemas de recomendación que se han identificado hasta el momento funcionan bajo el principio de búsqueda por temática y en algunos otros casos por perfiles de usuario, los cuales se asocian a estadísticas de búsqueda y resultados. Estas herramientas no ofrecen prestaciones de recuperación de información personalizada de acuerdo con las habilidades que posee cada usuario en el aprendizaje de lenguas extranjeras. Asimismo, no ofrecen una adaptación de los requerimientos del usuario al contenido específico de la base de datos, independientemente de cuál sea éste.

1.2. Estado del Arte

En este apartado se hace un recuento de métodos hallados en la literatura que son similares al presentado en este trabajo, señalando sus características más importantes así como aquellas que los distinguen.

Maryam Shokri [8] presenta un método basado en aprendizaje por refuerzo, el cual es personalizado dado que ayuda a los usuarios en la búsqueda de imágenes deseadas. Para interactuar con el usuario y presentar las muestras, el método aprende de las preferencias del usuario. En la fase de coincidencia, se comparan los IDs (identificadores) de los resultados apren-

didos con los IDs de los resultados de búsqueda y selecciona las imágenes personalizadas para presentarlas al usuario. Para aplicar aprendizaje por refuerzo a este proceso, la técnica de iniciativa mixta es aplicada para combinar retroalimentación subjetiva con resultados aprendidos para calcular la retroalimentación/recompensa para el agente. Una interfaz presenta las imágenes al usuario e interactúa con él para recibir retroalimentación del mismo.

En su trabajo [4], Mei-Hua Hsu presenta un sistema de recomendación en línea personalizado para el aprendizaje del inglés como segunda lengua. Este sistema ayuda a los estudiantes a seleccionar de forma apropiada lecciones que los ubique en sus intereses para lo cual utiliza un método de aprendizaje automático. El sistema pondera las lecciones disponibles en el recurso de enseñanza, de manera que si una lección tiene un peso alto, significa que los estudiantes de un grupo escogerán esta lección con mayor probabilidad. Con base en los experimentos realizados por los autores del método, la mayoría de los estudiantes aceptarán las recomendaciones de lectura que el sistema ofrece y es probable que su interés de leer las lecciones aumente significativamente.

Bhylenia Ríos [9] propone un sistema denominado 'tutor inteligente', diseñado como un recurso complementario para estudiantes en el ámbito de las lenguas extranjeras. Este sistema permite que los estudiantes avancen a su propio ritmo, participando en diversas actividades asociadas a las unidades básicas de aprendizaje. El 'tutor inteligente' opera mediante la técnica Multi-Agente, caracterizada por la interacción de varios agentes, cada uno abordando un problema mediante técnicas específicas. La resolución conjunta de estos agentes contribuye a la solución integral del problema. En términos generales, este sistema se enfoca en la recomendación de actividades relacionadas con el aprendizaje de la lengua inglesa.

Meiyun Gao expone un trabajo en [5], implementa mediante inteligencia artificial un método de recomendación personalizada para recursos en la enseñanza del inglés. Este método se basa en simular el comportamiento del usuario en búsquedas de recursos como audio y video en una base de datos. Obtiene palabras con alta frecuencia llamadas palabras claves; después, calcula la semejanza entre palabras claves y las características del usuario. Las características de usuario en este trabajo se dividen en dinámicas y estáticas. Las características dinámicas describen el comportamiento dinámico del usuario. Las características estáticas selecciona las preferencias del usuario y la información personal.

Los recursos que muestren alta semejanza son tomados como la recomendación objetivo. Este método mejora la precisión del contenido recomendado y la disponibilidad de los recursos para la enseñanza. Además, permite conocer mejor las necesidades individuales de los usuarios. En cuanto a funcionalidad, este sistema es el más similar al método que se presenta en este trabajo de tesis. Todos los trabajos mencionados en esta sección están al menos relacionados con el que se presenta en esta tesis. No obstante, hasta el momento no se encontró alguno que ofrezca temáticas y complejidad de vocabulario controladas para su presentación ante un usuario con requerimientos de nivel de dominio de la lengua que aprende. Tampoco se encontró alguno enfocado en la comprensión auditiva de la misma y que adapte las consultas del usuario al contenido de su base de datos.

1.3. Justificación

La comprensión auditiva se destaca como una de las habilidades más cruciales y, al mismo tiempo, una de las más desafiantes en el proceso de aprendizaje de la lengua inglesa [2]. Esta habilidad representa el 50 % del

tiempo usado para comunicarse [6], lo cual refleja la importancia que tiene para lograr una comunicación efectiva usando esta lengua.

Las grabaciones de audio usadas como recursos lingüísticos para mejorar la habilidad auditiva se encuentran divididos en niveles. Los niveles básicos (A1 y A2) presentan discursos claros y lentamente articulados para su mejor comprensión. Caso contrario sucede con las conversaciones de los niveles más avanzados (del B1 en adelante), en las cuales se presentan situaciones de la vida cotidiana, tal como lo establece el Marco Común Europeo de Referencia (MCER) para las lenguas [7]. El MCER señala que en la habilidad de comprensión auditiva, el alumno debe ser capaz de comprender los puntos principales de un discurso estándar, por ejemplo: asuntos familiares, situaciones laborales, escolares y de entretenimiento. Este mismo marco describe al nivel B1 como un nivel de supervivencia, donde se capacita al alumno para mantener relaciones sociales con hablantes nativos. Por ejemplo, la Universidad Tecnológica de la Mixteca (UTM) ofrece un programa obligatorio de inglés para sus más de 1500 estudiantes. Los niveles de inglés que se imparten toman como referencia su equivalente en el Marco Común Europeo [10], como son:

- KET= A2
- PET = B1
- FCE = B2

De acuerdo con los base de datos de la UTM [1], el 50 % de la comunidad estudiantil se encuentra cursando el nivel básico KET [11], mientras que otro 30 % asiste al nivel intermedio PET [12] y el 20 % restante se ubica en el nivel intermedio avanzado FCE [13].

[1] https://tinyurl.com/3e4wtwwc

Además, se ha observado que la habilidad de comprensión auditiva en todos los niveles es la de más alto porcentaje de reprobación entre los estudiantes [14]. El índice más alto de alumnos no aprobados se encuentra en el nivel intermedio PET-A y PET-B, que equivalen a los niveles B1 del MCER.

1.4. Hipótesis

Mediante la implementación de un agente neuronal como control de un método de recuperación de información a partir de una base de datos, es posible desarrollar un motor de búsqueda personalizado para la práctica del inglés, de modo que el agente adapte la consulta del usuario al contenido de la base de datos en cuanto a complejidad de vocabulario y temática.

1.5. Objetivos

Con base en el planteamiento del problema, se enuncia el objetivo general, los objetivos particulares y las metas que se trazaron al iniciar el proyecto.

1.5.1. Objetivo General

Desarrollar un motor de búsqueda personalizado para la práctica del inglés utilizando un agente neuronal como controlador del método de recuperación de información de una base de datos, de forma que el agente adapte la consulta del usuario al contenido de la base de datos en términos de complejidad de vocabulario y temática.

1.5.2. Objetivos Particulares y Metas

1. Crear la base de datos con guiones y audios a partir de la bibliografía de Cambridge para la enseñanza del inglés [11], [12] y [13] [2].

   a) Editar los guiones de las conversaciones de formato Word, así como descargar guiones y audios de [15] para almacenarlos en formato Excel.

   b) Describir los audios de las conversaciones en mp3.

2. Obtener los campos de la base de datos.

   a) Obtener los campos que describen a la base de datos mediante Python.

3. Implementar un entorno de aprendizaje por refuerzo, aplicando un sistema de recuperación de información como motor de búsqueda.

   a) Implementar método de recuperación de información (buscador).

   b) Adaptar la clase environment.

4. Implementar el agente neuronal (comunicarlo con el environment y con la señal de retroalimentación o recompensa).

   a) Implementar algoritmo por aprendizaje por refuerzo, para entrenar un agente neuronal (módulo de control), véase la figura 1.1.

5. Realizar experimentos de evaluación del sistema, en términos del rendimiento de aprendizaje.

[2] https://tinyurl.com/mwf4cnup

1.6. Metodología

La metodología de desarrollo del proyecto de investigación está dividido en las siguientes tareas:

1. **Creación de la base de datos:** Esta sección es fundamental porque representa la base de información con la que el sistema va a trabajar. La información contará con conversaciones, tomadas de los distintos textos bibliográficos y niveles con los que cuenta el Centro de Idiomas para la enseñanza de la lengua inglesa en la UTM. Mediante el programa Python se extraen las conversaciones del formato Word al formato Excel para su uso y procesamiento, teniendo como resultado la base de datos.

2. **Obtención de parámetros:** A partir de la base de datos, se tiene la posibilidad de captar todos los parámetros necesarios por medio del programa Python. Un ejemplo de los parámetros son la cantidad de palabras, la cantidad de vocabulario, el tiempo de duración de cada conversación, categorías relevantes como palabras de contenido y palabras funcionales; esta acción permite describir las palabras que contiene cada conversación en representaciones numéricas, que el agente entiende y las obtiene en sus parámetros de entrada.

3. **Implementación del entorno de aprendizaje por refuerzo:** Implementando un sistema de recuperación de información como motor de búsqueda e instalando la librería 'open aigym' para envolver nuestro motor de búsqueda, se logrará comunicar el sistema con el agente.

4. **Diseño del algoritmo:** La implementación del agente es mediante la librería 'stable-baselines for reinforcement learning'. La tarea del agente es realizar consultas al entorno mediante la señal de acción y

con la señal de retroalimentación recibirá del entorno una recompensa y un estado del sistema. El entorno define la recompensa, uno si lo que regresa el buscador y lo que solicita el usuario es cercano, cero en otro caso. El buscador analiza con base en la consulta del agente y devuelve al entorno las conversaciones mas similares en términos de similitud y complejidad. Estas conversaciones serán el estado actual del entorno y con las que el agente observará al sistema. Con esto, el trabajo del agente consiste en volver a realizar consultas sucesivamente hasta que aprende a formular consultas óptimas para los requerimientos del usuario [16]

5. **Implementación del sistema:** Después de terminar el diseño del algoritmo y conectar el agente con los demás bloques, se pasa a la fase de implementación para entrenarlo y ajustar las respuestas deseadas.

6. **Evaluación del sistema en términos de rendimiento del aprendizaje:** Se medirá el algoritmo en términos del vector de error. En otras palabras, esto busca encontrar la optimalidad mediante la función de Bellman y su objetivo es encontrar el óptimo estado mediante una acción óptima derivado de una política óptima.

En la figura 1.1 se observa el diagrama a bloques de cada uno de los módulos que componen la metodología de desarrollo:

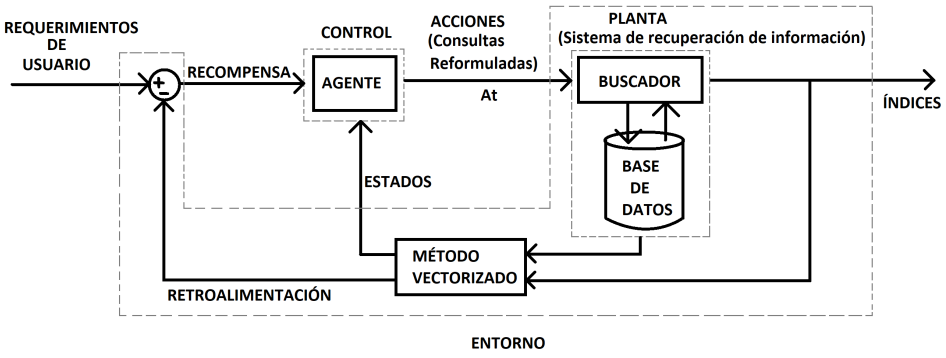


Figura 1.1: Diagrama a bloques.

En la figura 1.2 se muestra la metodología para el desarrollo del proyecto mediante un diagrama en vertical:

Figura 1.2: Diagrama vertical de la metodología.

Capítulo 2

Marco teórico

En este capítulo se brinda una descripción breve de los principales temas requeridos para el desarrollo del proyecto de tesis. En la primera sección se mencionan los desafíos del inglés para la adquisición de la habilidad auditiva como lengua extranjera. En la segunda sección se introducen los conceptos, así como la estructura, arquitectura y funcionamiento básico de una red neuronal. En la tercera sección se describe brevemente la relación que existe entre el algoritmo de aprendizaje por refuerzo y el esquema del control clásico. Así mismo, se explica el funcionamiento del aprendizaje por refuerzo aplicado en el agente. Un buscador convencional y las técnicas con las que operan como es el método TF-IDF se explican en la cuarta sección. La complejidad de vocabulario como parámetros de búsqueda en la base de datos se explica en la quinta sección. Por último, el algoritmo LDA y la biblioteca language tool se detallan en las secciones seis y siete respectivamente de este capítulo.

2.1. Retos de la comprensión auditiva del inglés como lengua extranjera

La comprensión auditiva se destaca como una habilidad fundamental en el proceso de aprendizaje de las lenguas, dado que constituye la destreza lingüística más empleada en la vida diaria. Esta habilidad desempeña un papel central en la comunicación y se refleja en el tiempo dedicado a la interacción, representando aproximadamente un 40 % a 50 % del mismo, mientras que el habla constituye entre un 25 % y 30 %, la lectura entre un 11 % y 16 %, y la escritura alrededor de un 9 % [6]. Por consiguiente, la comprensión auditiva es considerada ampliamente como uno de los desafíos más significativos para los estudiantes que aprenden inglés como lengua extranjera.

En los últimos años, ha habido un progreso significativo en el desarrollo de tecnologías de la información, como la inteligencia artificial, la computación en la nube y la minería de datos. Estas herramientas han tenido un impacto considerable en la implementación de la enseñanza asistida por medios multimedia. En respuesta a estos avances, muchos países desarrollados han realizado inversiones significativas en recursos humanos, materiales y tecnológicos en las escuelas, así como en la capacitación de los docentes. Según el estudio realizado por [17] sobre el diseño del aprendizaje asistido con recursos multimedia en estudiantes universitarios del área de inglés, se han identificado las siguientes ventajas:

- Provee una biblioteca de recursos multimedia para los estudiantes, de acuerdo con la clasificación multinivel como especializaciones y cursos.
- Los estudiantes pueden elegir materiales y formular planes de aprendizaje a partir de las características individuales, que puedan ayudar

a mejorar la eficiencia en su formación.

- Inspira el entusiasmo de los estudiantes por aprender, el contenido es muy vasto y el aprendizaje se vuelva más atractivo.

Se ha comprobado mediante diversos estudios que el uso de herramientas tecnológicas como apoyo para el desarrollo de la habilidad auditiva ha arrojado resultados positivos. Por ejemplo, un estudio realizado por [18] demostró que al utilizar cuentos digitales, los alumnos lograron mejorar su comprensión auditiva de manera significativa. Además, pudieron comprender mejor la estructura del idioma, ampliar su vocabulario y reconocer patrones sonoros, en contraste con una clase tradicional donde no se utilizó esta herramienta tecnológica.

En ese sentido, los desafíos asociados a las herramientas multimedia se centran en mejorar la capacidad de comprensión auditiva de los estudiantes, abordar problemas de pronunciación, facilitar el proceso de aprendizaje del idioma, optimizar el tiempo dedicado al estudio y fomentar un aprendizaje rápido del inglés. Según se menciona en el estudio realizado por [19], es fundamental ofrecer a los estudiantes una variedad de métodos de estudio y brindarles un entorno propicio para que puedan desarrollar de forma óptima sus habilidades en la lengua inglesa.

2.2. Redes Neuronales

Las actividades cognitivas complejas que el cerebro humano lleva a cabo están intrínsecamente relacionadas con el conocimiento adquirido a lo largo de la vida. Para afrontar tareas que, a menudo, desafían los enfoques algorítmicos tradicionales pero que se resuelven de manera natural para los seres humanos, se han desarrollado las Redes Neuronales Artificiales

(ANN, por sus siglas en inglés). Estas redes neuronales están inspiradas en la estructura y funcionamiento de las neuronas naturales presentes en el cerebro humano. La concepción de la primera neurona artificial se atribuye a los pioneros investigadores Warren McCulloch y Walter Pitts en 1943. Las ANN se destacan por su capacidad para aprender y adaptarse a través del tiempo, permitiéndoles resolver tareas de manera autónoma basándose en los datos que reciben [20].

Las neuronas naturales se componen de tres partes esenciales que desempeñan roles cruciales en la transmisión de la información. Las dendritas actúan como las entradas de la neurona, encargadas de recibir señales y estímulos del entorno. El núcleo, situado en el centro de la célula, asume la responsabilidad de procesar la información recibida a través de las dendritas. Por último, el axón funge como el canal de comunicación para transmitir la información procesada a otras neuronas o células del sistema.

Por otro lado, toda red neuronal artificial básica tiene 4 elementos básicos [21]. Estos elementos se muestran en la figura 2.1.

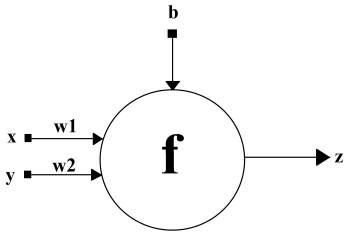


Figura 2.1: Elementos de una neurona artificial.

De acuerdo con la figura 2.1, el primer parámetro son las entradas 'x' e 'y', conocidas como dendritas, y su función principal es transmitir la información que ingresa a la neurona. Dentro de la neurona, los primeros elementos en recibir la información de entrada son los pesos 'w'. Estos pesos desempeñan un papel crucial en la neurona, ya que amplifican o reducen las señales de entrada según la naturaleza del problema a resolver. De hecho,

los pesos son considerados como el conocimiento de la neurona. Una vez que las señales de entrada son ponderadas por los pesos, se realiza una suma de todas las entradas. Para determinar si la neurona emitirá una señal de salida 'z', se emplea una función de activación *f*. Esta función se activa si la suma ponderada supera un umbral predefinido; de lo contrario, no habrá señal de salida. Para evitar que la salida sea cero cuando todas las entradas son cero, se introduce un parámetro adicional llamado bias 'b'con un peso definido, lo que evita este resultado. Finalmente, la señal de salida 'z' de la neurona se transmite a través de una dendrita hacia otra neurona.

En este modelo, la salida neuronal está dado por la ecuación 2.1:

$$z = f(w_1 x + w_2 y + b) \tag{2.1}$$

La función de activación f tiene como objetivo controlar la salida, es decir, regular su variabilidad y limitar su valor. La elección del tipo de función de activación dependerá del problema específico que se esté resolviendo.

Este funcionamiento específico de una neurona puede ser escalado, permitiendo la creación de una red neuronal. Esto implica que varias neuronas pueden interconectarse y colaborar en la ejecución de tareas más complejas. De acuerdo con [22] las redes neuronales se pueden estructurar de la siguiente manera:

- *Número de capas.* La cantidad de capas ocultas en una red neuronal se conoce como el número de capas, como se representa en la Figura 2.2.

  Una red neuronal puede estar compuesta por una sola capa oculta, lo que se conoce como una red *monocapa*, o también puede tener múltiples capas interconectadas, conocidas como una red *multicapa*. En

la práctica, añadir capas adicionales mejora la capacidad de la red neuronal para representar conocimiento más complejo.

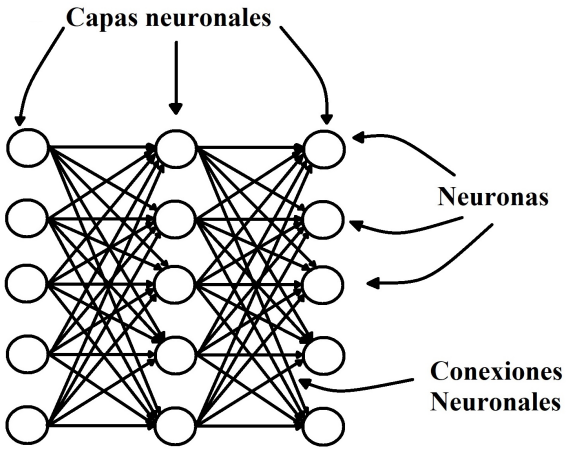


Figura 2.2: Capas de una red neuronal artificial. Fuente: tomado de [20].

- *El tipo de conexión.* Se refiere a si existe retroalimentación entre sus neuronas y capas. Cuando no hay ningún tipo de retroalimentación, se le llama *red neuronal no recurrente*. Por otro lado, cuando existe algún tipo de retroalimentación a través de lazos entre las neuronas, se le conoce como *red neuronal recurrente*.

- *El grado de conexión.* Cuando la salida de cada neurona está conectada a la entrada de todas las demás neuronas en la capa subsiguiente, se denomina *red neuronal totalmente conectada*. Por otro lado, si al menos una neurona no está conectada a la entrada de la neurona de la capa siguiente, entonces se clasifica como *red neuronal parcialmente conectada.*

  Si la información circula en un solo sentido, es decir, desde la capa de entrada hacia la capa de salida, entonces se denomina a este tipo de red neuronal como de *propagación hacia adelante o feedforward.* Por

otro lado, cuando existe retroalimentación entre las neuronas de la misma capa o entre capas anteriores, se le conoce como red neuronal de *propagación hacia atrás o feedback.*

Las redes neuronales aprenden ajustando las conexiones entre las neuronas, llamados pesos. Diferentes pesos provoca que la red produzca diferentes resultados para las mismas entradas. Así una red neuronal puede mejorar los resultados adaptando sus pesos de acuerdo con una regla de aprendizaje. El esquema general del aprendizaje se presenta en la figura 2.3:

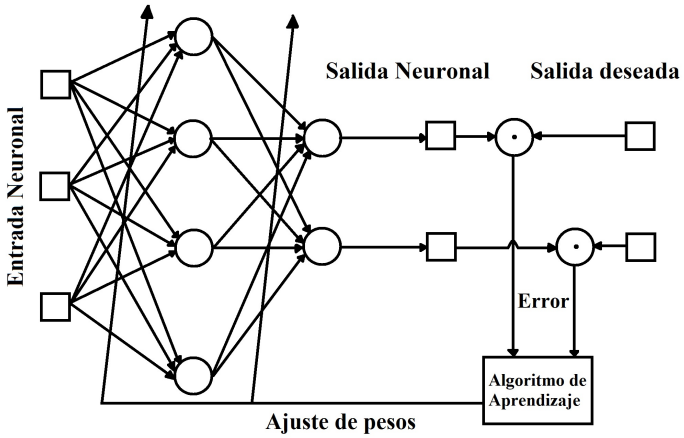


Figura 2.3: Esquema general de aprendizaje de una red neuronal. Fuente: tomado de [20].

El proceso de aprendizaje en las redes neuronales se ve fuertemente influenciado por la arquitectura de la red neuronal. En términos matemáticos, el objetivo es identificar los pesos ideales, representados como 'W', que minimicen la función de costo 'C(x, y)'. No obstante, en ocasiones, el proceso de aprendizaje no logra converger hacia un conjunto óptimo de pesos que cumpla con los criterios de aceptación. Para prevenir un entrenamiento perpetuo e ineficiente, se implementa una condición de paro conocida como 'done'. Esta condición se utiliza para evitar que la red neuronal continúe aprendiendo indefinidamente.

Mediante el método del gradiente, se derivan los valores óptimos que minimizan la función de costo, logrando así que esta converja hacia su valor mínimo. A continuación, la función de costo actualizará los pesos la red, disminuyendo el valor de la función de costo con los nuevos pesos, como se ilustra en la ecuación 2.2. 2.2.

$$W(k+1) = W(k) + \triangle W \tag{2.2}$$

Cuando una red neuronal aprende, recibe datos del entorno y adapta sus pesos según el objetivo. A estos datos se les conoce como conjunto de datos de aprendizaje. La palabra aprendizaje radica en el proceso de adaptar los pesos de la red neuronal para aprender a dar la respuesta deseada. Mientras la red está aprendiendo, hay un error entre las salidas objetivo (Y) y las salidas neuronales ($\hat{Y}$), como se muestra en la ecuación 2.3 :

$$e = Y - \hat{Y} \tag{2.3}$$

Debido a que el conjunto de datos de entrenamiento consta de múltiples valores, en cada iteración del proceso de entrenamiento de la red neuronal se obtendrá un error. En lugar de un vector de errores, se generará una matriz de errores. Esta matriz se obtiene mediante el cálculo del error cuadrático medio, que implica el promedio de la suma de los errores cuadráticos, como se muestra en la ecuación 2.4:

$$e_k = \frac{1}{2}\sum_{j}^{n}(Y_{kj} - \hat{Y}_{kj})^2 \tag{2.4}$$

Conforme avanza el proceso de aprendizaje, se espera que la red neuro-

nal produzca resultados que se asemejen cada vez más a las expectativas establecidas. Este acercamiento gradual hacia el cumplimiento de criterios de aceptación se mide en términos de 'épocas', que representan limitaciones en el número de iteraciones de aprendizaje. Por lo tanto, se considera que el proceso de aprendizaje llega a su fin cuando se satisface uno de estos dos criterios:

- Criterio de satisfacción: error total mínimo o distancia de peso mínima, según el paradigma del aprendizaje.
- Número máximo de épocas.

El proceso representado en la figura 2.3 se llama aprendizaje supervisado porque hay una salida deseada, pero las redes neuronales también pueden aprender de los datos de entrada sin ninguna salida deseada.

2.3. Aprendizaje por refuerzo

El concepto de Aprendizaje por Refuerzo (RL en inglés) constituye una de las tres principales áreas en el campo del Aprendizaje Automático, junto con el Aprendizaje Supervisado y el Aprendizaje No Supervisado. El aprendizaje por refuerzo representa un enfoque computacional fundamental para adquirir conocimiento a través de la interacción con un entorno. En este tipo de problemas, el objetivo es aprender qué acciones tomar, es decir, cómo asignar acciones a estados, con el fin de maximizar una señal de recompensa numérica. En cada problema de aprendizaje por refuerzo, se involucran los siguientes elementos [23]:

1. **Agente**. Este elemento se refiere al algoritmo o entidad que aprende

a través de las acciones que ejecuta en el entorno con el fin de lograr un objetivo específico.

2. **Entorno**. El entorno representa el sistema que se busca controlar a través de las acciones realizadas por el agente en las entradas del entorno. Las salidas del entorno se traducen en estados y recompensas que son proporcionados al agente.

3. **Política**. La política establece el comportamiento del agente en un instante dado. En términos generales, es un mapeo que relaciona los estados percibidos del entorno con las acciones que el agente debe tomar en esos estados.

4. **Señal de recompensa** La señal de recompensa define el objetivo en un problema de aprendizaje por refuerzo. En cada paso, el entorno proporciona al agente una recompensa. El principal objetivo del agente consiste en maximizar la acumulación de recompensas a lo largo del tiempo. La señal de recompensa, por tanto, determina qué eventos son beneficiosos o perjudiciales para el agente.

5. **Modelo del entorno** El modelo del entorno es una representación que imita el comportamiento del entorno, o en términos más generales, permite realizar inferencias sobre cómo se comportará el entorno.

El Aprendizaje por Refuerzo surgió como una solución a los desafíos planteados por los Procesos de Decisión de Markov (MDP en inglés). Estos MDP pueden concebirse como secuencias de eventos, donde la probabilidad de que ocurra un evento en particular depende del evento inmediatamente anterior o la probabilidad de que ocurra un evento futuro depende del evento actual. De manera gráfica, un MDP se representa comúnmente como una cadena de Markov, como se ilustra en la figura 2.4:

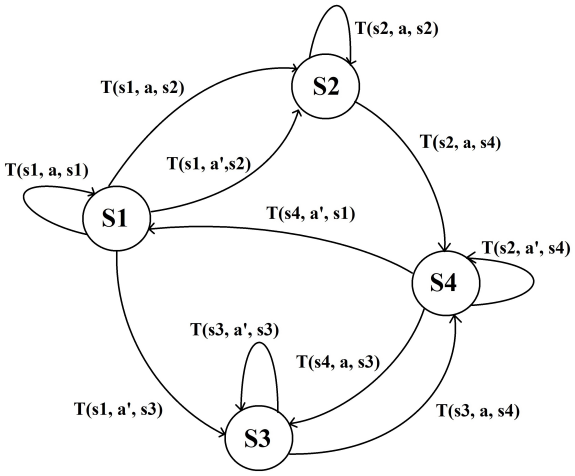


Figura 2.4: Representación gráfica de un proceso de Markov sencillo.

En esta representación gráfica del proceso de Markov se observan cuatro estados identificados como s1, s2, s3 y s4. Además, se muestran dos acciones posibles (a, a'). La función T se utiliza para representar la probabilidad de transición de un estado a otro, considerando una acción específica. Cada flecha que conecta estados en el gráfico de transición también está asociada a una recompensa particular. Esta estructura gráfica permite visualizar cómo la transición entre estados se lleva a cabo, teniendo en cuenta las acciones tomadas y las recompensas correspondientes en cada caso. En términos simples, un sistema puede considerarse completamente observable a través de un Proceso de Decisión de Markov, lo que significa que se establece una relación que vincula la interpretación de una observación con los estados que caracterizan al entorno. [24].

Estos procesos se caracterizan por tener un espacio de estados S y un espacio de acciones A. Cumplen con ciertas propiedades, incluyendo la existencia de una probabilidad p(s',r|s,a) para la transición del estado s $\in$ S al estado s' dado una acción a $\in$ A, y esta transición está influenciada por la política π(s) que se sigue. En otras palabras, para cada estado s, se define una política π(a|s) que especifica la acción a tomar.

Además, en cada transición de estado, se relaciona una función de recompensa $r(\mathrm{s}, \pi(\mathrm{s}))$ que se busca maximizar. A partir de estos elementos, surgen los valores V, que se definen a través de la ecuación de Bellman [23]. Esta ecuación desempeña un papel fundamental en la formulación de estrategias y la toma de decisiones en estos procesos.

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{2.5}$$

donde γ representa el factor de descuento, el cual desempeña el papel de priorizar las recompensas adquiridas en un menor número de transiciones (o en un tiempo más breve).

No obstante, la resolución de un MDP mediante los valores de estado V puede resultar engorroso y complicado, especialmente porque en aplicaciones del mundo real, cuando un agente se enfrenta a un entorno por primera vez, rara vez se tiene un conocimiento completo del espacio de estados, y mucho menos de las probabilidades de transición entre ellos. Fue en este contexto que surgió el algoritmo Q-Learning [25], el cual modifica la ecuación de Bellman con el propósito de encontrar los valores Q asociados a cada par acción-estado, sin requerir un conocimiento previo de la función T. Estos valores Q se calculan mediante un método iterativo, como se detalla a continuación:

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_a(Q(s_{t+1}, a)) \tag{2.6}$$

donde α corresponde al factor de aprendizaje, y r_t representa la recompensa obtenida en la transición en el tiempo t. En resumen la ecuación 2.5 sirve para estimar cuanta recompensa se obtendrá siguiendo una política determinada en ese estado particular. Por otro lado, la función 2.6 evalúa la acción a tomar en un estado determinado siguiendo un política especifica

con el propósito de lograr la mayor recompensa posible. El propósito del aprendizaje por refuerzo es encontrar una política óptima para todo par acción y estado que maximice la recompensa acumulada en todo el tiempo establecido.

En su esencia, el aprendizaje por refuerzo se caracteriza por ser un tipo de problema de lazo cerrado, donde las acciones tomadas por el sistema tienen influencia en su estado futuro. Más allá de identificar cuáles acciones conducen a la máxima recompensa, la característica más intrigante radica en el hecho de que las acciones pueden impactar no solo la recompensa inmediata, sino también el estado subsiguiente y, a través de esto, a todas las recompensas posteriores. Por lo tanto, la particularidad distintiva de los problemas de aprendizaje por refuerzo incluye la característica del lazo cerrado, la falta de instrucciones directas sobre qué acciones tomar y las consecuencias de las acciones tomadas.

La Figura 2.5 ilustra de manera general el proceso de funcionamiento de un algoritmo de aprendizaje por refuerzo.

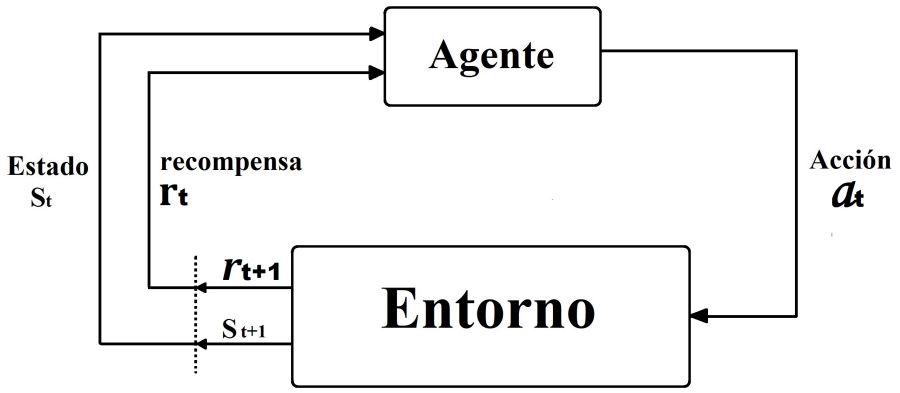


Figura 2.5: Representación del proceso de funcionamiento de un algoritmo de aprendizaje por refuerzo.

De acuerdo con la ecuación 2.5 y como se observa en la figura 2.5, a cada par acción A_t y estado S_t le corresponde un nuevo estado S_{t+1} con

una recompensa asignada r_{t+1}, teniendo una recompensa acumulada de $r_t + r_{t+1}$.

Como se puede apreciar en la figura 2.5, el proceso de funcionamiento del algoritmo de aprendizaje por refuerzo opera dentro de un sistema de lazo cerrado. Esto implica que la señal resultante del proceso se compara constantemente con una señal de referencia con el propósito de mantener un seguimiento continuo de la evolución de la variable que se busca controlar. Este enfoque de bucle cerrado se asemeja a los sistemas de control clásico en ingeniería.

2.3.1. Del control clásico al aprendizaje por refuerzo

Bajo este concepto de lazo cerrado, se establece una relación entre el funcionamiento del algoritmo de aprendizaje por refuerzo y el circuito de lazo cerrado utilizado en el ámbito de los sistemas de control clásico. En la figura 2.6, se presentan los diagramas de estos dos sistemas para ilustrar dicha relación.

Ambos diagramas funcionan bajo los siguientes conceptos:

- **Sistema a controlar.** En ambos diagramas existe el sistema a controlar, en control clásico es llamado planta y en RL se conoce como entorno.
- **Señal de referencia.** En el sistema de control clásico, la señal de entrada de referencia es r y en aprendizaje por refuerzo, la referencia se encuentra dentro del entorno.
- **Señal de retroalimentación.** Ambos sistemas integran la retroalimentación, característica de los circuitos de lazo cerrado. En control

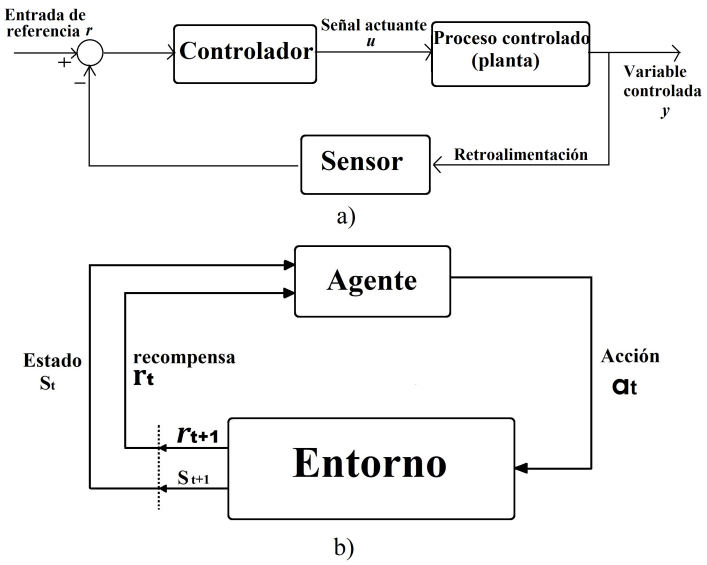


Figura 2.6: (a) Representación del diagrama de control clásico y (b) Representación del diagrama de aprendizaje por refuerzo.

clásico, la señal va de la salida al sumador donde se compara con la referencia y después hacia el controlador. En aprendizaje por refuerzo, la señal va del entorno, donde se compara, hacia el agente.

- **Control.** El control sobre la planta o entorno lo realiza un controlador llamado así en control clásico y en RL se denomina agente.
- **Señal actuante.** La señal de acción a_t o actuante u es la que se emite desde el control (agente) hacia el sistema a controlar (entorno).

En control clásico, el objetivo es minimizar el error entre la señal de referencia y la señal de salida [26], mientras que en el diagrama de aprendizaje por refuerzo el objetivo es maximizar el total de recompensas que recibe a largo plazo entre la señal de referencia y la señal de acción.

Como se puede observar, ambos sistemas son de lazo cerrado y los bloques principales comparten en esencia las mismas funciones. Ahora, los objetivos a largo plazo que persiguen estos mismo son inversamente proporcional, esto quiere decir, mientras que en control clásico la meta es

minimizar el error, en RL el objetivo es maximizar la recompensa. En un sentido inverso, esto nos dice que si en los sistemas RL la recompensa es mínima, en control clásico el error será máximo.

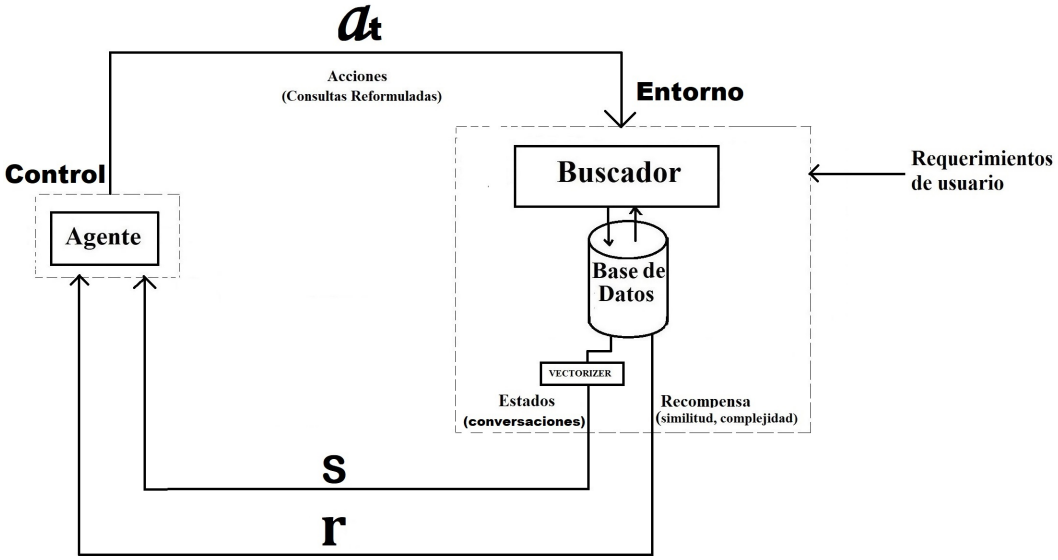


Figura 2.7: Transformación del sistema de control clásico hacia el sistema aprendizaje por refuerzo.

En la figura 2.7, se muestra el diagrama del buscador de conversaciones con la transformación de la estructura inicial como sistema de control clásico hacia la estructura final como sistema de aprendizaje por refuerzo. La transformación de estos sistemas se puede observar claramente mediante las líneas punteadas, donde los bloques principales del control clásico se conservan. El primero que permanece es el bloque planta llamado en RL entorno. En este bloque entorno se encapsulan los bloques buscador, base de datos y un método importante llamado *vectorizer*. El segundo bloque que permanece es el bloque control, en RL este bloque es llamado agente. Otra parte importante que se conserva es la señal de retroalimentación, en RL esta señal queda en función de la recompensa y los estados. Por último, la señal actuante en RL se transforma en señal de acciones.

Bajo este esquema de transformación final, el diagrama de aprendizaje

por refuerzo, basándose en el objetivo de recompensas máximas a lo largo del tiempo, intentará que el agente aprenda a tomar mejores decisiones. Para interés de este proyecto, tomar mejores decisiones se traduce en realizar mejores consultas desde el agente hacia el entorno. Así mismo, las consultas realizadas estarán determinadas por políticas que dependen a su vez del algoritmo RL a implementar.

2.3.2. Tipos de algoritmos

Dependiendo del enfoque con el que se quiera trabajar existen distintas clasificaciones en los algoritmos RL. Sin embargo, una de las clasificaciones más comunes en los algoritmos RL se da en dos contextos. El primero son los algoritmos diseñados para entornos sin modelos definidos. El segundo son los algoritmos diseñados para entornos que trabajan con un modelo establecido.

Los algoritmos que trabajan con entornos que no tienen un modelo definido ofrecen mas ventajas que los que trabajan con un modelo establecido. De acuerdo con [27], los algoritmos que interactúan con entornos sin modelo se clasifican como se muestra en la figura 2.8.

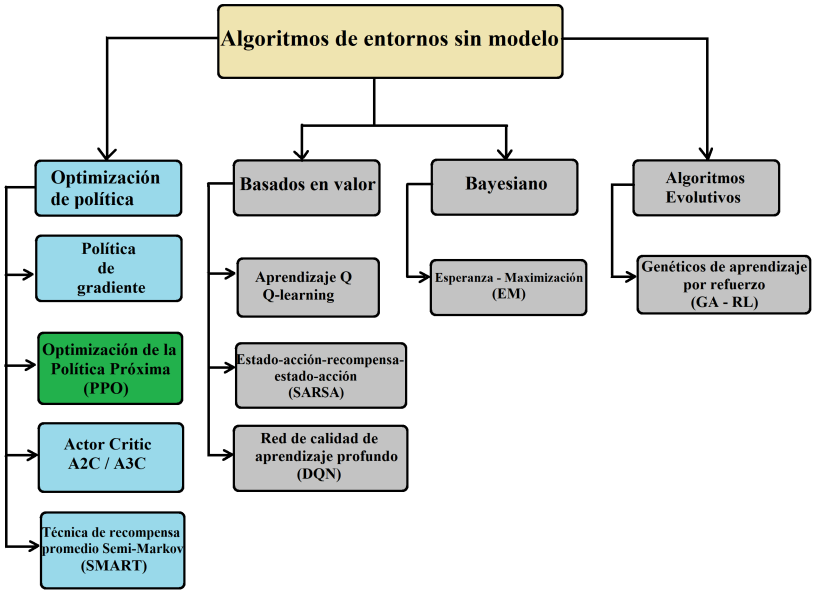


Figura 2.8: Clasificación de algoritmos RL.
Fuente: tomado de [27].

La clasificación que se muestra en la figura 2.8 muestra dos importantes clasificaciones. La primera es la clasificación de algoritmos con optimización de política y la segunda son los algoritmos basados en valor. Para interés de este proyecto solo se mencionará de forma breve los algoritmos basados en valor y algunos de optimización de política, dando mayor énfasis a los de política de gradiente y en particular el algoritmo PPO.

Algoritmos basados en valor

Los algoritmos basados en valor se dedican a estimar y optimizar dos funciones de valor distintas. La primera de ellas estima y optimiza el valor del estado próximo al cual el agente puede llegar, es decir, evalúa cuán beneficioso resulta transitar del estado actual del agente a un estado futuro, independientemente de la acción tomada. La evaluación del valor del estado se realiza en términos de recompensas positivas. Por otro lado, la segunda función de valor se encarga de evaluar cuán ventajoso es el par estado-

acción disponible para el agente. En otras palabras, determina qué tan remunarable es, en términos de recompensas positivas, elegir una acción específica para llegar a un estado futuro determinado. Algunos ejemplos de algoritmos basado en valor son los siguientes:

1. *Q-learning clásico* es un algoritmo diseñado para determinar una política óptima para un agente que opera en un entorno discreto y estocástico. Su principal objetivo es maximizar el valor esperado de la recompensa total al elegir la acción más adecuada en cada situación. En otras palabras, la meta de Q-learning es aprender la función Q (véase la ecuación 2.6). Esta función Q evalúa qué tan buena es la acción a seleccionar en términos de recompensas positivas en un estado específico [25]. La información proporcionada por Q-learning al agente le ayuda a tomar decisiones más acertadas y, como resultado, a obtener recompensas más favorables a lo largo del tiempo.

   Algunas ventajas de este algoritmo son que no requiere un modelo del entorno, es conocido por converger al valor óptimo para la función Q con suficiente exploración y bajo ciertas condiciones, lo que lo hace confiable para aprender políticas a largo plazo. Las desventajas de este algoritmo son que en espacios de estado o acciones continuas, Q-learning puede ser menos eficiente. Además, si el entorno es dinámico y cambia con el tiempo, el modelo puede volverse obsoleto.

2. *Deep Q-Networks*(DQN) representa una mejora significativa del algoritmo *Q-learning clásico*. La mejora radica en la sustitución de la tabla o matriz de valores utilizada por *Q-learning* para almacenar los valores de estados y acciones, mediante la implementación de una red neuronal profunda. El algoritmo *Q-learning* demuestra eficacia en problemas relativamente pequeños, donde la cantidad de estados y acciones no es considerablemente grande. Sin embargo, cuando la com-

plejidad del problema aumenta, la tabla de *Q-learning* puede volverse poco práctica.

Para abordar este inconveniente, se introdujo el algoritmo DQN, que reemplaza la tabla de valores con dos redes neuronales. La primera red neuronal, conocida como la *red neuronal principal*, tiene la función de estimar los valores actuales basándose en el estado y la acción. La segunda red neuronal, denominada *red neuronal objetivo*, se encarga de aproximar los valores futuros de estado y acción. El proceso de aprendizaje se lleva a cabo en la *red neuronal principal*, mientras que la *red neuronal objetivo* actualiza sus parámetros después de un cierto número de iteraciones. Cuando se llega a un número determinado de iteraciones, la *red neuronal principal* transfiere sus parámetros a la *red neuronal objetivo*, con el propósito de mejorar las predicciones de esta última. Este enfoque contribuye a la estabilidad y eficacia del aprendizaje en entornos más complejos.

Aunque son varias las ventajas que ofrece el modelo DQN, podemos destacar la capacidad de manejar entornos con espacios de estados grandes y complejos. Además, permite generalizar sobre estados similares, lo que lo hace más eficiente en términos de memoria. Por el otro lado, algunas desventajas que presenta DQN son durante el entrenamiento porque presentan problemas de estabilidad. También, si no se administra bien la política epsilon-greedy, el agente puede dejar de explorar lo suficiente y quedarse atascado en una política subóptima.

3. SARSA*(State-Action-Reward-State-Action* representa una variante del clásico algoritmo Q-learning. SARSA es un algoritmo RL basado en política, en el sentido de que la acción es ejecutada según la política actual para determinar el valor de Q [27].

   Converger hacia políticas más seguras y estables, asegurar una buen

manejo entre los procesos de exploración y explotación son algunas de las ventajas del algoritmo SARSA. Por otra parte, existen algunas desventajas, como el converger mas lento hacia políticas óptimas ya que las actualizaciones se basan en la política actual. También, la taza de aprendizaje α y el factor de descuento γ puede afectar la velocidad de convergencia y la calidad de la política aprendida.

Algoritmos basados en optimización de política

Como lo menciona [28], hay siempre al menos una política que es mejor o igual que todas las demás políticas, esto es una política óptima. Los algoritmos basados en optimización de política son los que trabajan para mejorar una estrategia y con base en esto poder realizar mejores acciones. La estrategia se representa mediante un conjunto de valores que deciden la próxima acción a seguir. A este conjunto de valores se les conoce como política.

A diferencia de los algoritmos basados en valor, que evalúan estados o pares acción-estado para optimizar las recompensas a lo largo del tiempo, los algoritmos basados en la optimización de política directamente evalúan la política para determinar la acción más beneficiosa en términos de recompensa en un estado particular. En otras palabras, los algoritmos de optimización de política definen una función de política $\pi(a|s)$ que ayuda a decidir qué acción tomar en un estado específico.

A continuación se mencionan algunos algoritmos basados en optimización de política:

1. Política de gradiente, se enfoca en mejorar dos parámetros clave en los algoritmos RL. En los algoritmos que emplean una función de valor, el objetivo principal es reducir el error entre las predicciones

de los valores y los valores reales. Este propósito se logra al disminuir la función de error mediante el descenso de gradiente, ajustando los parámetros del algoritmo en la dirección opuesta al gradiente de la función de error.

En contraste, en los algoritmos de política, el objetivo principal es maximizar la recompensa a lo largo del tiempo. Este proceso implica ajustar los parámetros del algoritmo en la dirección del gradiente de la función de rendimiento. En otras palabras, a través del ascenso de gradiente, se busca encontrar la política óptima que maximice la recompensa a lo largo del tiempo.

Aplicar el algoritmo de política de gradiente en conjuntos de datos muy grandes y encontrar soluciones óptimas en menor tiempo son algunas ventajas que ofrece este algoritmo, por mencionar algunas. Por otra parte, inicializar de forma incorrecta los parámetros de la política puede provocar que queden atrapados en mínimos locales o converjan a políticas subóptimas, siendo esto una desventaja.

2. Advantage Actor-Critic (A2C), es un algoritmo síncrono aplicado en aprendizaje profundo y en RL, empleando dos redes neuronales para captar patrones o modelos a partir de los datos existentes en el entorno. La primera red neuronal, llamada 'actor', tiene la función de decidir la acción a implementar en un estado específico. En cambio, la segunda red neuronal, conocida como 'critic', se encarga de estimar la función de valor en ese estado, es decir, de calcular el valor que tiene esa acción en ese estado con respecto al valor real obtenido [29]. La diferencia entre estos dos valores se conoce como ventaja (advantage) y proporciona información al agente para estimar acciones más efectivas. El objetivo principal de este algoritmo es maximizar el número de recompensas obtenidas del entorno a lo largo del tiempo a través

de mejores acciones ejecutadas por el agente.

La sincronización en la actualización de las dos redes neuronales implica que ambas ajustan simultáneamente sus parámetros (polític a y valores). Este enfoque garantiza una mayor velocidad y estabilidad durante el proceso de aprendizaje del algoritmo.

3. La Optimización de Política Próxima (PPO) es un algoritmo empleado en el ámbito de aprendizaje profundo y Aprendizaje por Refuerzo. Este algoritmo opera mediante la arquitectura actor-critic, donde la red neuronal 'actor' desempeña un papel crucial al establecer y perfeccionar la política para tomar acciones precisas y efectivas en relación con el entorno. Simultáneamente, la red neuronal 'critic' evalúa el estado en términos de obtener las mejores recompensas positivas [30].

   En la red neuronal 'critic' se calcula el valor del estado actual en términos de recompensas a futuro. Esta evaluación se realiza mediante la función de valor (2.7):

$$V_{\phi}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s] \tag{2.7}$$

   donde:

   - $\mathbb{E}$: es la expectativa de retorno, en otras palabras, es la cantidad de recompensa que se espera acumular en el futuro al elegir ese estado.
   - γ: es el factor de descuento para dar una ponderación a las recompensas a largo plazo.
   - r_t: es la recompensa recibida en el tiempo t.
   - s: es el estado actual.

La función desempeñada por la red 'critic' es crucial para determinar la importancia que tiene ese estado en términos de utilidad en el futuro. Después de calcular el valor del estado actual, se procede a calcular la importancia de esa acción específica en ese estado en particular. Para obtener esta información, se realiza el cálculo de la 'función de ventaja' (2.8):

$$\hat{A} = Q_\phi(s_t, a_t) - V_\phi(s_t) \tag{2.8}$$

donde:

- $Q_\phi(s_t, a_t)$: es la estimación de retorno a partir de tomar una acción a en el estado s, ecuación (2.6).
- $V_\phi(s_t)$: es el valor del estado s siguiendo la política actual, ecuación (2.7).

Según la 'función de ventaja' (2.8), cuando la diferencia entre la función de valor Q y la función de valor V es positiva, se interpreta que la acción del agente fue mejor de lo esperado. En contraste, si la diferencia es negativa, se concluye que la acción del agente fue peor de lo esperado. Para la red 'critic', el agente se orienta mediante la métrica del error entre la ventaja y el valor estimado por la red (2.9):

$$L_{critic} = \frac{1}{2}\sum_t (\hat{A}_t - V_\phi(s_t))^2 \tag{2.9}$$

donde:

- $\hat{A}_t$: es la ventaja en el tiempo t.
- $V_\phi(s_t)$: es el valor del estado s en el tiempo t.

La función (2.9), también conocida como error cuadrático medio (MSE, por sus siglas en inglés), indica que cuando la ventaja es negativa, el

error tiende a aumentar; en cambio, si la ventaja es positiva, el error tiende a disminuir. Esto sugiere que las acciones del agente mejoran a medida que la ventaja se vuelve positiva.

Por otro lado, la red neuronal 'actor' tiene la responsabilidad de encontrar la política óptima para el agente, aquella que le brinde el mayor rendimiento en términos de recompensa. Una característica distintiva de este algoritmo es que no realiza modificaciones sustanciales en términos de ajustar su política; por el contrario, efectúa los ajustes de manera gradual. Este enfoque evita cambios bruscos al actualizar la política, contribuyendo así a proporcionar una mayor estabilidad en el proceso de aprendizaje.

Para realizar el proceso de optimización en la política de manera gradual en el agente, se requiere de un parámetro crucial conocido como la 'función de error'. Esta función guiará al agente en su objetivo de encontrar su política óptima. La función de error del agente PPO se muestra en 2.10:

$$L(\theta) = \mathbb{E}[min(r_t(\theta)\hat{A}, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A})] \tag{2.10}$$

donde:

- (θ): Son los parámetros de la política.
- $r_t(\theta$: Representa la relación de probabilidad entre la política nueva y la política antigua y se define como $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{antigua}(a_t|s_t)}$, donde el númerador $\pi_\theta(a_t|s_t)$ indica la probabilidad de elegir la acción a_t en el estado s_t con la política actual que contiene los parámetros θ. Por otro lado, el denominador $\pi_{antigua}(a_t|s_t)$ representa la selección de la misma acción de acuerdo con la política antigua.
- $\hat{A}_t$: Es la '*función de ventaja*' en el tiempo t. Este parámetro

indica, en términos de rendimiento, cuánto mejor es la acción tomada en comparación con el promedio de acciones en ese estado.

- *clip*: El término '*clip* es una métrica que evita cambios drásticos en la política, promoviendo así una mayor estabilidad en el aprendizaje del agente.
- ϵ : Es un hiperparámetro utilizado para controlar los rangos de actualización de la política.

La función de error del agente $L(\theta)$ se optimiza por medio de la técnica de descenso de gradiente. De esta manera, la red neuronal 'critic' ajustara sus pesos hasta converger.

La principal ventaja de PPO radica en su capacidad para mejorar la política de manera estable y controlada, evitando que el aprendizaje se vea afectado adversamente. No obstante, es importante señalar la presencia de hiperparámetros significativos, como el tamaño del lote y la tasa de aprendizaje, que deben ajustarse con precisión para optimizar su rendimiento. Además, su implementación puede resultar más compleja en comparación con otros algoritmos.

2.3.3. Sistema CartPole

El problema del Cartpole es uno de los ejemplos clásicos dentro del campo de aprendizaje por refuerzo. Este es un ejemplo de un sistema dinámico no lineal el cual se utiliza para probar diferentes algoritmos dentro del aprendizaje automático. El objetivo del problema es aplicar una fuerza horizontal (derecha o izquierda) a un carro (cart) para mantener en posición vertical la barra (pole) que se encuentra sujeta a un eje en la parte superior del carro y tratar de evitar que la barra se caiga. Al hecho de que la barra se caiga se le conoce como condición de paro *done* dentro del campo de

aprendizaje por refuerzo, esto quiere decir, que el sistema falló al tratar de conseguir su objetivo. En la figura 2.9 se ilustra el sistema Cartpole.

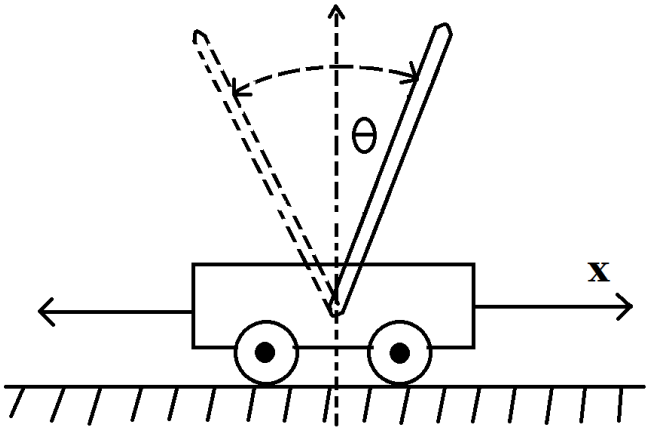


Figura 2.9: Sistema Cartpole.

Se va a considera condición de paro *done* cuando la barra cae después de rebasar un ángulo (etiquetado como θ) desde la vertical o si el carro rebasa una cierta distancia horizontal (etiquetada como x) en ambos lados [31]. La barra vuelve a la posición vertical después de cada condición de paro *done*. En la Eq. (2.11) se muestra la ecuación para el *done* en el CartPole:

$$done = \text{bool}[(x < \text{-}x\_thd) \vee (x > x\_thd) \vee (\theta < -\theta\_thd) \vee (\theta > \theta\_thd)] \tag{2.11}$$

donde:

x: Es la distancia horizontal que puede recorrer el carro.

$\text{-}x\_thd$: Es el umbral negativo de la distancia x.

x_thd: Es el umbral positivo de la distancia x.

θ: Es el ángulo que recorre la barra antes de caer.

$-\theta\_thd$: Es el umbral negativo del ángulo θ.

θ_thd: Es el umbral positivo del ángulo θ.

Para este sistema, la recompensa podría ser uno en cada paso en el cual la condición de paro *done* no ocurrió, en otro caso, si hay un *done* podría regresar un cero. El objetivo del aprendizaje por refuerzo es maximizar esa recompensa y se obtendrá siempre y cuando la barra se mantenga en balance.

2.3.4. Evaluación de algoritmos RL

En el proceso de entrenamiento de algoritmos de aprendizaje por refuerzo (RL), es frecuente la aparición de ruido cuando el agente está en fase de exploración. Por tanto, al evaluar estos algoritmos, es esencial llevar a cabo ciertas tareas. Siguiendo las pautas propuestas por [32] para la evaluación de algoritmos RL, se deben seguir algunas recomendaciones:

- Es esencial proporcionar al algoritmo un entorno de prueba independiente para evaluar el rendimiento del agente en momentos específicos.
- Se recomienda realizar evaluaciones periódicas del agente utilizando un número definido de episodios de prueba (n generalmente se encuentra entre 5 y 20) y calcular el promedio de la recompensa por episodio para obtener una estimación precisa.
- En el caso de los algoritmos PPO y A2C, que vienen configurados de forma predeterminada con una política estocástica, se debe intentar probar los algoritmos con una política determinista para comparar sus rendimientos.
- Analizar la curva de recompensa promedio proporciona una valiosa

referencia para comprender el comportamiento del agente a lo largo del tiempo.

2.4. Recuperación de información

2.4.1. Buscadores

Cuando se tiene claro qué información se busca de un tipo específico o sobre un tema particular, un motor de búsqueda puede resultar una herramienta sumamente eficaz. En la actualidad, los motores de búsqueda específicos de dominio ganan popularidad gracias a su mayor precisión y funcionalidad [33].

El término 'personalizado' implica la construcción de un perfil de interés del usuario mediante el análisis del historial de búsquedas y los clics realizados. El motor de búsqueda realiza diversas tareas, como entrenar un modelo de búsqueda personalizado, capturar de manera secuencial los registros del historial de búsqueda, construir perfiles de usuario de forma dinámica y aprender estrategias de clasificación de información, ya sea en formato de audio, vídeo o documentos. En última instancia, el buscador crea un modelo adaptado a los cambiantes intereses de los usuarios.
En la figura 2.10 se muestra el funcionamiento de un buscador personalizado básico.

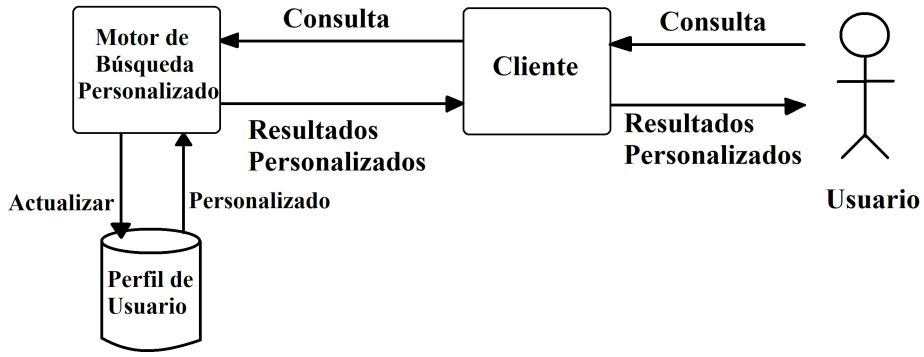


Figura 2.10: Diagrama básico de un buscador personalizado.

2.4.2. Tf-idf y similitudes

Para obtener la representación de documentos en una base de datos, se emplea la 'Matriz de términos-documentos', que se construye mediante el método TF-IDF (por sus siglas en inglés, Frecuencia de Término - Frecuencia de Documento Inversa). Esta técnica tiene como objetivo medir la frecuencia con la que aparece un término o frase en un documento específico y compara esta frecuencia con el número total de documentos que mencionan ese término en toda la colección de documentos [34].

La Matriz de términos-documentos está compuesta por filas que representan términos en todos los documentos, y las columnas consisten en identificadores de documentos. Cada elemento de la matriz, llamado celda, representa la frecuencia de cada palabra ponderada por algún número. Cada vector fila en la matriz corresponde a un documento y se utiliza para calcular la similitud con una consulta dada. Esta similitud facilita la recuperación de documentos de la matriz que resulten ser los más similares a la consulta, ofreciendo un método eficaz y sencillo de recuperación de información.

El método TF-IDF se divide en dos partes. La primera parte (TF) cal-

cula la frecuencia del término en un documento. La segunda parte (IDF) evalúa si un término es común o no en la colección de documentos. Su objetivo es reducir el peso de los términos que aparecen con mucha frecuencia en todos los documentos y aumentar el peso de los términos que aparecen pocas veces. Matemáticamente, este método se explica de la siguiente manera.
La frecuencia de término (TF) es la frecuencia del término (t) en el documento (d), como se muestra en la ecuación 2.12:

$$tf_{t,d} = count(t, d) \tag{2.12}$$

La frecuencia del término se puede representar de forma escalada como logaritmo de base 10, por lo que el número se vuelve más pequeño y el proceso de cálculo se vuelve más rápido. También, se agrega uno para que no que exista el log 0, como se muestra en la ecuación 2.13:

$$tidf_t = \log_{10}(count(t, d) + 1 \tag{2.13}$$

Enseguida, se obtiene la IDF dividiendo el número total de documentos N por el número de documentos que contienen el término y se toma el logaritmo de ese cociente, como se muestra en la ecuación 2.14:

$$idf_t = \log_{10} \frac{N}{df_t} \tag{2.14}$$

Luego, TF-IDF se calcula como la ecuación 2.15:

$$Tfidf(t, d, N) = tf(t, d) \times idf(t, N) \tag{2.15}$$

El método TF-IDF afecta el valor en cada celda. Elimina todas las palabras que se muestran con frecuencia en los documentos, pero al mismo tiempo que no son importantes, como 'y', ó', íncluso', etc. Con base en eso,

se usa esto como el valor en cada celda de la matriz.
Con la base de datos transformada en una matriz de documentos vectorizada, se puede realizar una recuperación de información mediante un tema de usuario vectorizado (q), esto se consigue por medio de una similitud. Para calcular la similitud entre el tema de usuario con la matriz de documentos, se usa la fórmula del coseno, como se muestra en la ecuación 2.16:

$$cosine(q, d) = \frac{q * d}{qd} \tag{2.16}$$

El método cosine calcula la similitud de dos vectores, mediante el producto escalar dividido por la multiplicación de la longitud de cada vector. El valor del coseno oscila entre [-1,1]. Debido a que no tiene valores negativos, se ignora el valor negativo porque nunca sucede. Con estos parámetros de similitud se identifican los documentos más cercanos al tema de interés del usuario.

2.5. Complejidad Textual

El desarrollo de la habilidad de comprensión de lectura es un proceso que evoluciona a lo largo de las distintas etapas del crecimiento humano. Esta destreza es crucial en la vida de las personas, ya que les posibilita adquirir un mayor conocimiento de su entorno. No obstante, algunos factores pueden obstaculizar el adecuado desarrollo de esta habilidad por parte del lector. Uno de estos factores es la complejidad textual, la cual se refiere a la cantidad y dificultad de la información presentada en un texto. Estas dificultades se determinan mediante la longitud y complejidad de las oraciones, la frecuencia de palabras complicadas y la estructura general de la información. Estas métricas pueden ser útiles para clasificar textos en

diferentes niveles de complejidad, permitiendo a la persona o estudiante adaptarse según su nivel de dominio en el tema o habilidad específica en la lectura.

2.5.1. Complejidad de vocabulario

Los procesos que conforman el complejo mecanismo de la comprensión lectora han sido objeto de interés de múltiples disciplinas que buscan una aproximación al tema. Entender las causas que dificultan la lectura implica dar cuenta de distintas variables que pueden afectar el desempeño en la comprensión. Si el lector reconoce inmediatamente el significado de las palabras que lee, puede concentrarse en la comprensión del texto. Esto se debe a que para interpretar el significado de un texto, es necesario tener conocimiento de alrededor del 90 % al 95 % de las palabras presentes en él [35].

Por lo anterior, el estudiante debe reconocer al instante la mayoría de las palabras o expresiones de un texto para comprenderlo. En ese sentido, la técnica para obtener la complejidad de vocabulario de un texto o documento es mediante la normalización, la cual se muestra en la ecuación 2.17.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.17}$$

donde:
X': el valor normalizado en el conjunto de datos.
X: el valor en el conjunto de datos.
X_{min}: el valor mínimo en el conjunto de datos.
X_{max}: el valor máximo en el conjunto de datos.

2.6. Algoritmo LDA para temas preponderantes

jsLDA es una biblioteca de JavaScript que incorpora el algoritmo de Latent Dirichlet Allocation (LDA, por sus siglas en inglés) para llevar a cabo el modelado de temas directamente en el navegador. El algoritmo LDA es un modelo generativo probabilístico empleado en el procesamiento del lenguaje natural para identificar temas predominantes en un conjunto de documentos [36]. A continuación, se presenta un resumen general de la operación del algoritmo LDA, que jsLDA implementa:

1. Preparación de los datos. Antes de aplicar el algoritmo LDA, es crucial que los documentos cumplan con un formato específico. Esto implica transformar los documentos en una matriz documento-término, donde cada fila representa un documento y cada columna representa un término en el vocabulario. Se emplea el método TF-IDF para calcular las ponderaciones y la frecuencia de los términos en cada documento.

2. Inicialización de parámetros. El siguiente paso implica la inicialización de los parámetros del algoritmo, tales como el número deseado de temas preponderantes y las asignaciones iniciales de palabras aleatorias a los temas de cada documento.

3. Iteración entre pasos. El algoritmo sigue un ciclo iterativo que ajusta los temas y la asignación de palabras a los temas. En el primer paso, estima la distribución de temas en cada documento y la distribución de palabras en cada tema mediante técnicas de inferencia estadística. En el segundo paso, actualiza la asignación de palabras a los temas basándose en las distribuciones de temas y palabras estimadas en la etapa anterior. Estos pasos se repiten hasta que el usuario decida detener el proceso o hasta que se alcance la convergencia en los parámetros

del modelo.

4. Convergencia del algoritmo. Cuando el algoritmo alcanza la convergencia, se obtienen las distribuciones de temas en los documentos y las distribuciones de palabras en los temas. Estas distribuciones son útiles para identificar los temas más destacados en los documentos, lo que posibilita la realización de diversas tareas analíticas y exploratorias.

Mediante la inferencia, LDA busca identificar las distribuciones de temas y palabras que mejor explican los documentos presentes en el corpus. Vale la pena destacar que LDA es un algoritmo no supervisado, lo que implica que no depende de etiquetas previas de temas en los documentos; en cambio, identifica automáticamente estos temas a partir de patrones encontrados en los datos.

2.7. Librería language tool python para errores gramaticales

La librería language_tool_python, de acuerdo con [37], es un corrector ortográfico y gramatical aplicable a textos en varios idiomas. En el caso del inglés, aplica una amplia variedad de reglas gramaticales, las cuales cubren diversos aspectos gramaticales. Estas reglas han sido diseñadas para identificar y corregir errores comunes en textos en inglés. Algunas de estas reglas gramaticales junto con ejemplos se describen a continuación según [38] y [39].

De acuerdo con [37], la librería language_tool_python es un corrector ortográfico y gramatical que se aplica en textos y en distintos idiomas. Esta librería aplica una amplia variedad de reglas gramaticales en la lengua

inglesa. Estas reglas abarcan varios aspectos de la gramática y se han desarrollado para detectar y corregir errores comunes en textos que están en inglés. Algunas de las reglas gramaticales que language_tool aplica en la lengua inglesa se mencionan a continuación con sus respectivos ejemplos [38] y [39].

1. **Concordancia de género y número.** Se refiere a la coincidencia apropiada entre sustantivos, adjetivos, pronombres y verbos en términos de género y número para lograr una oración gramaticalmente correcta y coherente. Un ejemplo de error de concordancia de género se muestra a continuación:

   Oración incorrecta: "He is a nurse, but she are a doctor."

   En esta oración, se identifica un error de concordancia de género en el verbo "are". El pronombre "she" se refiere a una mujer, por lo que el verbo debería estar en la tercera persona del singular, que es "is", para concordar correctamente con "she".

   Oración corregida: "He is a nurse, but she is a doctor."

   En la oración corregida, el verbo "is" se ha conjugado correctamente para concordar con el pronombre "she" y mantener la concordancia de género adecuada.

2. **Tiempos verbales incorrectos.** Estos errores son aquellos en los que el verbo no se utiliza en la forma gramaticalmente adecuada para expresar el momento en el que ocurre una acción. Los tiempos verbales en inglés indican cuándo sucede una acción: pasado, presente o futuro. Utilizar el tiempo verbal incorrecto puede afectar la precisión y comprensión del mensaje en una oración. La siguiente oración es un ejemplo de error de tiempo verbal:

   Oración incorrecta: "I am try not to eat chocolate this week."

En esta oración, se detecta un error en la conjugación del verbo "try". Cuando estamos hablando de algo que está sucediendo en el momento, debemos agregar la continuidad al verbo para no caer en errores de tiempos verbales.

Oración corregida: "I am trying not to eat chocolate this week."

Con el verbo "try" conjugado en la forma continua la oración ya cobra sentido y coherencia.

3. **Uso incorrecto de preposiciones.** Estos errores se refiere a la mala selección de preposiciones al conectar palabras o frases dentro de una oración. Las preposiciones son palabras pequeñas pero significativas que ayudan a expresar la relación espacial, temporal o lógica entre diferentes elementos de una oración. A continuación se menciona un ejemplo de error en el uso incorrecto de preposiciones.

   Oración incorrecta: "I'll visit you at Friday night."

   En esta oración, se visualiza un error en el uso de la preposición "at", porque la preposición "at" solo se utiliza para hacer referencia a horas o momentos específicos, y para referirnos a la noche. Para hacer referencia a días de la semana se utiliza la preposición "on".

   Oración corregida: "I'll visit you on Friday night."

   Se menciona que la librería language_tool es de código abierto, lo que implica que periódicamente se agregan nuevas reglas gramaticales.

Capítulo 3

Desarrollo del proyecto

En la primera sección de este capítulo, se presenta el funcionamiento general del sistema buscador personalizado, destacando el algoritmo de aprendizaje por refuerzo a través de un diagrama de flujo. La segunda sección detalla la estructura y contenido de la base de datos que alberga las conversaciones. En la tercera sección se centra en la descripción de la clase 'Buscador' incluyendo sus atributos y métodos principales. La cuarta sección aborda el funcionamiento completo del sistema de recuperación, detallando la clase 'Environment' con sus métodos y atributos. Finalmente, la quinta sección describe la implementación del agente neuronal junto con los parámetros con los que opera.

3.1. Funcionamiento general del sistema

Considerando los enfoques de control clásico y aprendizaje por refuerzo que se detallan en la sección 2.3 y la subsección 2.3.1, respectivamente, se explica de forma general el funcionamiento del sistema de búsqueda personalizado. Con el objetivo de brindar una visión más amplia del algoritmo,

la figura 3.1 presenta el diagrama de flujo del sistema, dividido en tres etapas: la creación de la clase "Environment", el proceso de aprendizaje del agente y las predicciones realizadas utilizando el modelo óptimo.

Para construir el módulo entorno, se inicia definiendo las clases "Environment" y "Buscador" con sus respectivos atributos y métodos. Posteriormente, se crean instancias de ambas clases: el objeto del environment lo etiquetaremos como *obj_entorno*, mientras que el objeto del buscador lo etiquetaremos como *obj_buscador*. Estos dos objetos están vinculados, lo que significa que el *obj_buscador* se pasa como parámetro al *obj_entorno*. Esta integración de objetos constituye el núcleo del módulo entorno. Desde la perspectiva del control clásico, el módulo entorno actúa como una planta, mientras que desde la óptica del aprendizaje por refuerzo, opera como un entorno. Asimismo, este módulo entorno va a interactuar en todo momento con el agente.

El funcionamiento del sistema se inicia cuando el usuario ingresa dos datos importantes: el primero es el tema de interés, que etiquetaremos como *tema_usr*, y en segundo, el nivel de complejidad de vocabulario que etiquetaremos como *complej_usr*. Una vez que estos dos datos han sido ingresados, el *obj_entorno* los procesa e inmediatamente genera una consulta que etiquetaremos como *query*. Este *query* se envía al *obj_buscador*.

Por su parte, el *obj_buscador* recibe el *query* y procede a compararlo con las conversaciones presentes en la base de datos. Este proceso de comparación será etiquetado como *similitud*. Utilizando esta medida de *similitud* como base, el *obj_buscador* envía nuevamente al *obj_entorno* tanto el *vector de conversaciones* como el *vector de complejidades*.

Regresando al *obj_entorno*, obtenemos su estado, el cual lo etiquetamos como *vector de observaciones*, a través de la media del *vector de conversa-*

ciones. Respecto a la recompensa, la que se etiqueta como *rwd*, se calcula de la siguiente manera: si el *vector de conversaciones* y el *vector de complejidades* presentan una *similitud* más alta con *tema_usr* y *complej_usr*, respectivamente, la *rwd* será igual a uno. Por el contrario, si esta *similitud* es baja, la *rwd* será cero.

Luego de esta evaluación, el *obj_entorno* transmite el estado al agente, compuesto por la *rwd* obtenida y un *vector de observaciones*. Posteriormente, el agente procede a evaluar estos valores y ajusta inmediatamente sus parámetros, con el fin de efectuar una nueva consulta al *obj_entorno*.

El agente utiliza la retroalimentación proporcionada por el *vector de observaciones* y la *rwd* para orientarse, con el propósito de saber cómo fueron evaluadas las consultas anteriores al *obj_entorno*. Este proceso de retroalimentación permite al agente perfeccionar las consultas realizadas a lo largo del tiempo, mejorando su rendimiento de manera gradual.

El agente es una instancia de la Clase del tipo de algoritmo a implementar y se asigna a una variable llamada *modelo*. El *modelo* recibe dos argumentos: el primero es el tipo de *política* que se implementará y el segundo es el *obj_entorno*. El proceso de aprendizaje del *modelo* lo realiza a través del método *learn*. Dentro de los múltiples parámetros que acepta el método *learn*, uno de ellos es el número de pasos, referido como *n_steps*. Este parámetro indica cuántas consultas el agente realizará en la base de datos. Así, al concluir el número establecido de pasos, se espera que el agente haya acumulado la máxima cantidad de recompensas durante todo el proceso. A este proceso de aprendizaje del agente se le denomina entrenamiento.

Una vez que el agente ha completado su proceso de entrenamiento, la etapa final implica realizar predicciones de consultas en una nueva base de

datos. Este proceso se logra al cargar el *modelo*, el *obj_ entorno* y el *vectorizer* que se utilizaron y guardaron durante el entrenamiento. Finalmente, mediante un bucle *for* el agente entra en interacción con el *obj_ entorno*, que incorpora la nueva base de datos. En este *obj_ entorno*, el agente buscará y obtendrá las conversaciones más afines en términos de *tema_ usr* y *complej_ usr*.

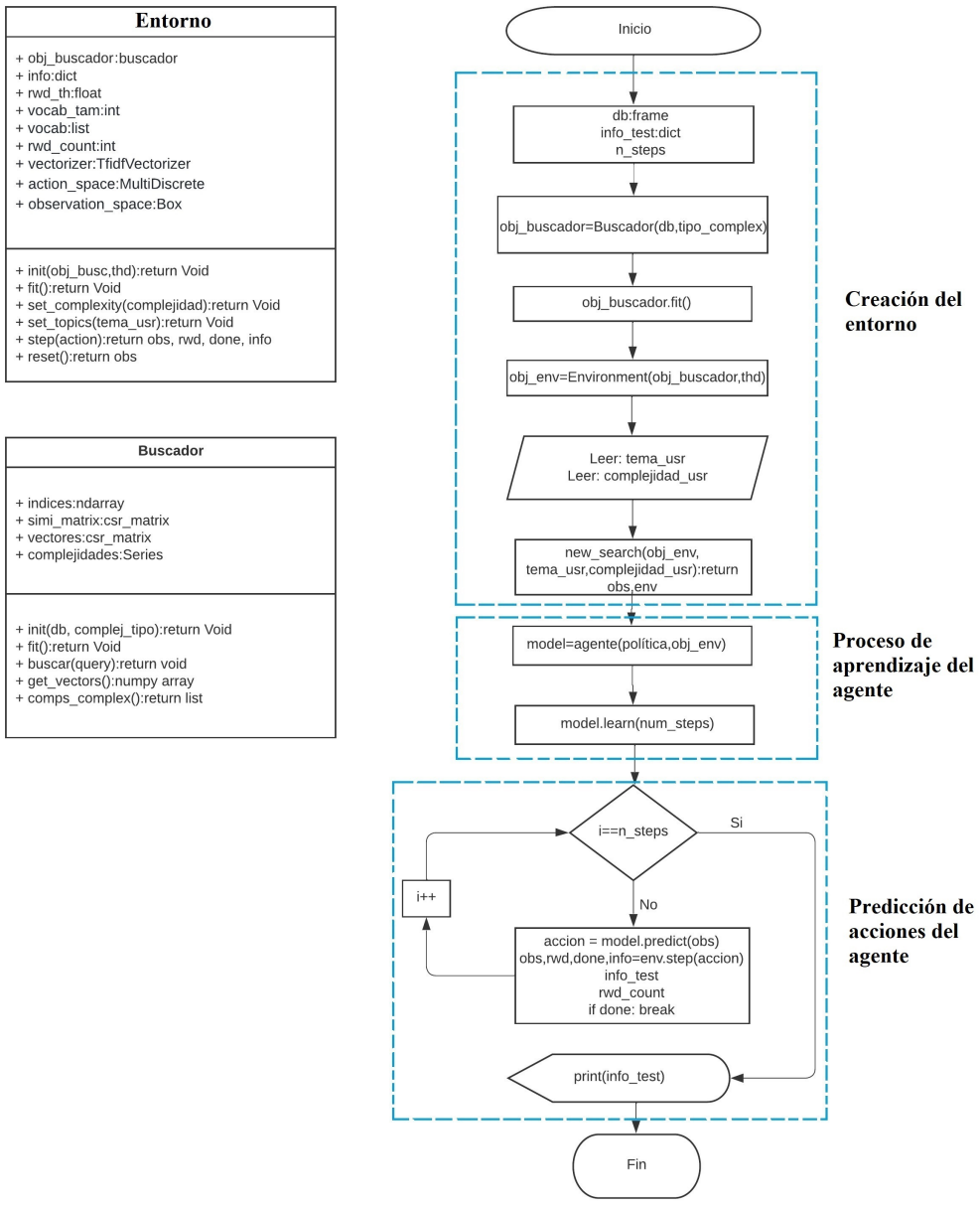


Figura 3.1: Diagrama de flujo del sistema buscador personalizado.

3.2. Módulo base de datos del sistema

La base de datos del sistema buscador personalizado abarca todas las conversaciones obtenidas de libros de texto y fuentes de acceso libre en internet. Estas conversaciones comprenden diferentes niveles, desde el básico hasta el avanzado. Con el fin de construir esta base de datos en formato Excel, se desarrolló un algoritmo mediante el lenguaje de programación Python. El diagrama de flujo de este algoritmo se muestra en la figura 3.2.

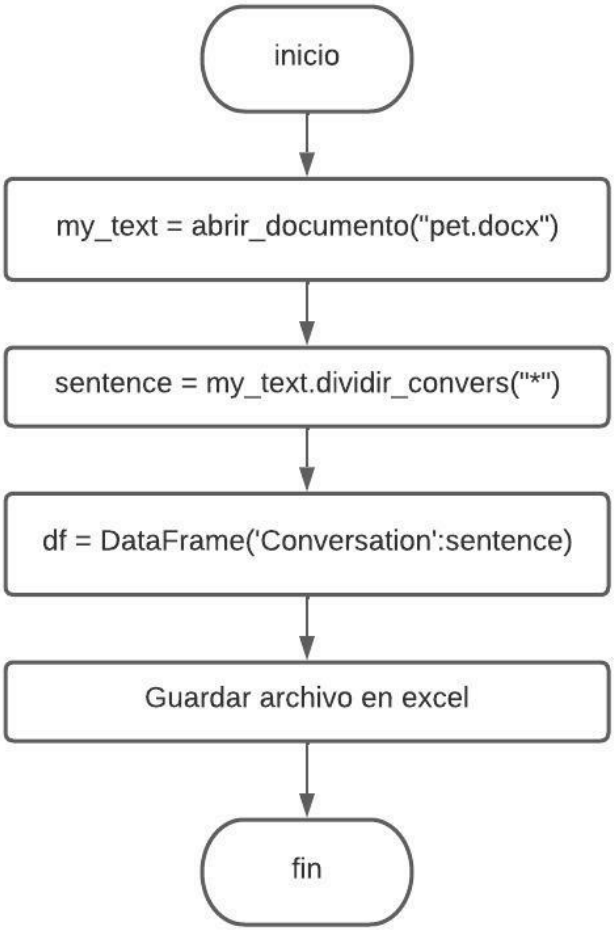


Figura 3.2: Diagrama de flujo para la obtención de la base de datos.

El funcionamiento del diagrama de flujo de la figura 3.2 comienza por

la lectura del archivo Word que contiene todas las conversaciones. Cada una de estas conversaciones está delimitada por el carácter '*', que indica su finalización. A continuación, se extraen las conversaciones individuales del archivo. Luego, estas conversaciones se incorporan en un archivo de Excel, siendo identificadas en dicho documento por la columna 'Conversation'. Finalmente, se guarda el archivo que almacena la totalidad de las conversaciones, conformando así la base de datos.

3.3. Módulo buscador

Antes de comenzar a explicar el funcionamiento de la clase Buscador, es necesario señalar que los diagramas de clases en el lenguaje de modelado unificado (UML) se componen de tres partes fundamentales: el nombre de la clase, así como los atributos y métodos que esta clase contiene, tal como se ilustra en la figura 3.3.

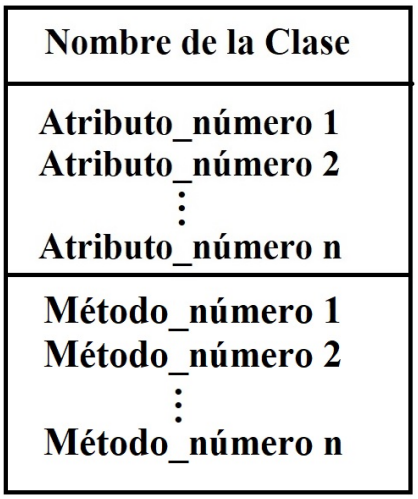


Figura 3.3: Estructura de una clase en UML.

Como se ha mencionado anteriormente, el módulo buscador será una instancia de la clase Buscador y que se etiquetó como *obj_ buscador*. La

tarea principal de la clase Buscador es devolver a la clase Environment el *vector de conversaciones* y el *vector de complejidades*. Estos dos vectores que regresa la clase Buscador serán los más similares al *query* generado por el agente. A continuación, en la figura 3.4 se observan los atributos y métodos más importantes de la clase Buscador.

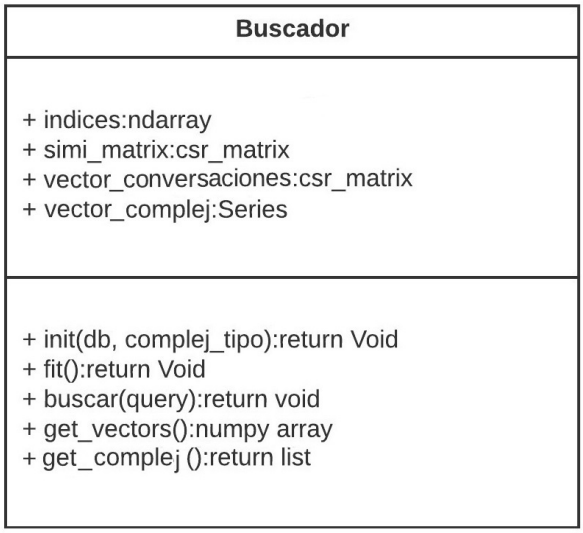


Figura 3.4: Clase Buscador.

Los atributos esenciales de la clase Buscador son los siguientes:

- *vector de conversaciones*: contiene las conversaciones extraídas de la base de datos.
- *vector de complejidades*: contiene las complejidades de vocabulario de las conversaciones correspondientes.
- *indices*: almacena los índices de las conversaciones que el buscador devuelve.
- *simi_matrix*: aloja las similitudes entre el *query* y las conversaciones en la base de datos.

- *vectorizer*: representa una instancia de la clase TfidfVectorizer en Python para encontrar la matriz de documentos X, esto conforme a la Eq.2.15.

Los métodos sobresalientes de la clase Buscador son los siguientes:

- *init*: se utiliza para inicializar los atributos de la clase.
- *fit*: ajusta la base de datos por medio del objeto *vectorizer* para obtener la matriz de documentos X. Este objeto posee el método *vocabulary_keys*, el cual permite extraer el vocabulario presente en la matriz. Al conocer el vocabulario, se obtiene automáticamente el tamaño del mismo, que se identifica como *vocab_tam*.
- *buscar*: recibe como parámetro al *query* del agente y por medio de *vectorizer* lo transforma en un vector de frecuencia de palabras etiquetado como *query_vect* conforme a la Eq.(2.13). Mediante la métrica *cosine_similarity* se encuentran las similitudes entre *query_vect* y la matriz de documentos X, estas similitudes junto con sus índices se almacenan respectivamente en las variables *simi_matrix* e *indices*.
- *get_vectors*: obtiene las cinco conversaciones más cercanas a *query*.
- *get_complej*: obtiene las complejidades de las cinco conversaciones más cercanas a *query*.

3.4. Módulo entorno

Como se mencionó anteriormente, el módulo entorno será una instancia de la clase Entorno y esta se etiquetó como *obj_entorno*. Sus funciones principales son recibir el *query* y regresar dos parámetros al agente: el primero es

la *rwd* y el segundo son las observaciones llamadas *vector_ observaciones*. A continuación se observa en la figura 3.5 los atributos y métodos más importantes de la clase Entorno.

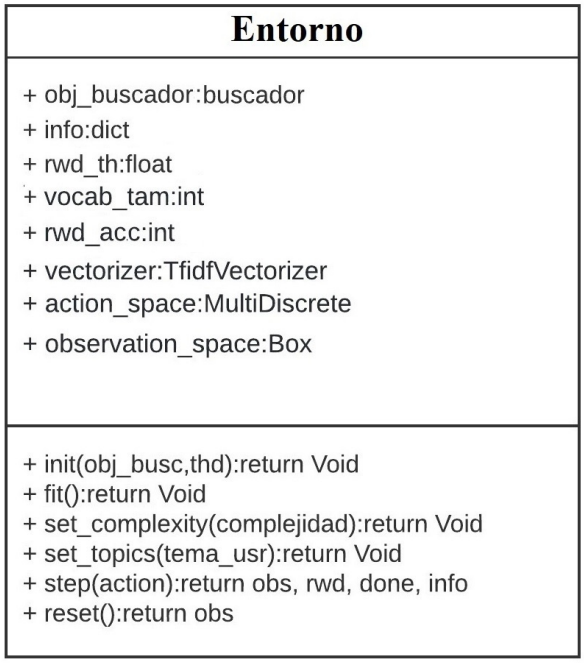


Figura 3.5: Clase Entorno

Los atributos mas importantes de la clase Entrono son los siguientes:

- *obj_ buscador*: es de tipo objeto y va a contener al objeto buscador.
- *info*: es de tipo diccionario y va a tener la información de salida del entorno.
- *s_ thd*: es de tipo flotante y es el umbral de la *similitud*.
- *c_ thd*: es de tipo flotante y es el umbral de la complejidad de vocabulario.
- *rwd_ thd*: es un atributo es de tipo flotante y es el umbral de la recompensa.

- *rwd*: es de tipo entero y va a contener las recompensas que el módulo entorno envíe al agente.
- *rwd_acc*: es un atributo de tipo entero y su tarea es almacenar la recompensa acumulada por episodio.
- *vector de observaciones*: es de tipo array y va a contener las observaciones del módulo entorno.
- *vectorizer*: este atributo es recuperado desde la clase Buscador, es de tipo Tfidfvectorizer y la función que desempeña en la clase Environment es para vectorizar el *tema_usr*. Esta vectorización la asignaremos a una variable que se etiqueta como *tema_usr_vect*.
- *vocab_tam*: este atributo va a contener el tamaño de vocabulario de la base de datos, es de tipo entero y es recuperado desde la clase Buscador.
- *action_space*: atributo de tipo espacio de acciones multidiscreto, tiene cuatro argumentos de tamaño *vocab_tam* para dimensionar las consultas del agente.
- *observation_space*: atributo de tipo espacio de observaciones, tiene una estructura de datos tipo diccionario y de tamaño *vocab_tam*.

Los métodos más destacados de la clase Environment son los siguientes:

- *init*: es un método para inicializar los atributos de la clase Entorno. Este método recibe dos argumentos: el primero es el *obj_buscador* y el segundo argumento es el umbral de la recompensa *rwd_thd*.
- *set_complexity*: método para insertar la *complej_usr*
- *set_topic*: método para insertar el *tema_usr*.

- *reset*: método donde se inicializa al *obj_ entorno* en sus valores iniciales, después manda un *query* aleatorio al método *step* para comenzar el proceso de búsqueda con los parámetros descritos anteriormente. Regresa como parámetro el *vector de observaciones*.

- *step*: método donde se realizan las tareas fundamentales del *obj_ entorno* como el calculo de la *rwd* y el *vector de observaciones*. En seguida, se explica el funcionamiento de este método.

En términos generales, las tareas del método *step* comienzan tomando el *query* del agente y los datos de interés del usuario, es decir, *tema_ usr* y *complej_ usr*. Con la ayuda del *obj_ buscador*, este conjunto de información se procesa y es devuelto nuevamente al *obj_ entorno* en forma de un *vector de conversaciones* y un *vector de complejidades*.

Estos dos vectores son procesados por el *obj_ entorno* dentro de su propio método *step*, obteniendo como resultado final el *vector de observaciones* y la *rwd*. Ahora bien, es importante profundizar en la descripción detallada del proceso que conduce a la obtención de estos dos resultados.

Es importante tener presente que en los algoritmos de Aprendizaje por Refuerzo, los estados son una representación de la información que describe la situación actual en la que se encuentra un agente en su entorno. Para este proyecto, se implementan ajustes específicos con el propósito de calcular medidas con tendencia central.

Concretamente, el *vector de observaciones* se establece en función de la *media* del *vector de conversaciones*, así como de la *mediana* tanto del *vector de complejidades* (etiquetado como *complej_ mediana*) como del vector de *similitudes* (etiquetado como *simi_ mediana*). El vector de *similitud* se obtuvo mediante la métrica *cosine_ similarity*, al comparar el *vector de*

conversaciones y el *tema_usr_vect*.

Como se puede observar, el *vector de observaciones* guarda una relación directa con dos parámetros fundamentales: *simi_mediana* y *complej_mediana*. De acuerdo con los procesos de Markov que se abordan en la sección 2.3, menciona la probabilidad de que ocurra un evento depende del evento inmediato anterior. Con base en este principio de Markov, se adaptará los argumentos que conforman el estado del entorno.

Por lo tanto, el *vector de observaciones* debe considerar los valores previos que ocurrieron en el *obj_entorno*. Esto implica que se deben incluir el valor anterior de *simi_mediana* (etiquetada como *simi_mediana_anterior*) y el valor previo de la *complej_mediana* (etiquetada como *complej_mediana_anterior*). De esta forma, se adapta el *vector de observaciones* para que sea coherente con los principios de dependencia temporal característicos de los procesos de Markov.

Adicionalmente, es esencial que el agente sea capaz de observar la evolución de estas variables en cada uno de los estados en los que se encuentre. Con este fin, se introduce una medida de cambio (Δ) para cada una de estas variables. En la eq. 3.1 se puede apreciar cómo se realiza el cálculo de Δ_simi:

$$\Delta\_simi = simi\_mediana - simi\_mediana\_anterior \tag{3.1}$$

De manera análoga, se calcula $\Delta\_complej$ tal como se muestra en la Eq. 3.2:

$$\Delta\_complej = complej\_mediana - complej\_mediana\_anterior \tag{3.2}$$

Estos valores de Δ se agregan como dos parámetros adicionales al *vector*

de observaciones. Para determinar la medida de tendencia central del *vector de observaciones*, se obtiene su vector promedio, tal como se observa en la Eq. 3.3:

$$obs = (vector\ de\ observaciones + vector\ de\ observaciones\ anterior)/2 \tag{3.3}$$

Hasta este punto hemos logrado obtener el *vector_observaciones*, uno de los dos parámetros que el *obj_entorno* envía al agente.

Como se menciona en la sección (2.3), el siguiente parámetro a obtener es la *rwd*. Para calcular la *rwd*, es necesario que el episodio no llegue a su fin, esto quiere decir, que la condición de paro llamado *done* no se active. En la Eq. 3.4 se puede observar que el *done* se determina mediante una condición booleana:

$$done = |(complej\_usr - complej\_mediana)| => c\_thd \vee simi\_mediana <= s\_thd \tag{3.4}$$

La Eq. 3.4 establece que la variable *done* será verdadera si al menos una de las dos condiciones es verdadera. Esto quiere decir, si en la primera condición el valor absoluto de la diferencia entre la *complej_usr* y la *complej_mediana* es mayor o igual al umbral *c_thd*, o si en la segunda condición, la similitud mediana es menor o igual al umbral *s_thd*. En otro caso, si la variable *done* es falsa, se procede a calcular la *rwd* para el *query* que realizó el agente.

Para calcular la *rwd*, es necesario realizar algunos cálculos preliminares. En la Eq. 3.5 se observa el cálculo de una pre-recompensa etiquetada como *pre_rwd*:

$$\begin{gathered} \text{diff} = |(complej\_usr - complej\_mediana)| \\ anti\_complej = 1 - diff \end{gathered}$$

$$pre\_rwd = (anti\_complej + simi\_mediana)/2 \tag{3.5}$$

En el cálculo de la variable *pre_rwd*, el escenario óptimo ocurre cuando la *complej_usr* y la *complej_mediana* son iguales, lo que resulta en que el valor de la variable 'diff' sea igual a cero. Esto lleva a que la variable *anti_complej* tome el valor de uno. En consecuencia, la variable *pre_rwd* tiende a uno en este caso. Por otro lado, cuando *complej_usr* y *complej_mediana* tienden a ser distantes, el valor de 'diff' tiende a uno, lo que hace que el valor de *anti_complej* tiende a cero. Por lo tanto, en este escenario, la variable *pre_rwd* tiende a cero. En otras palabras y de forma resumida, se puede decir que si la *complej_mediana* es cercana a la *complej_usr*, el valor del *pre_rwd* va a tender a 1, en caso contrario, si son muy opuestos, va a tender a 0.

Continuando con el calculo del *rwd*, se procede a comparar los valores del *pre_rwd* con el umbral de recompensa denominado *rwd_thd*. Primero se verifica si *pre_rwd* es mayor que *rwd_thd*. En caso afirmativo, el valor de *rwd* será igual a 1, como se ilustra a continuación:

$$Si \quad pre\_rwd => rwd\_thd$$
$$rwd = 1$$

En caso contrario, si *pre_rwd* es menor a *rwd_thd*, entonces el valor de *rwd* será igual a 0, como se muestra a continuación:

$$Si \quad pre\_rwd < rwd\_thd$$
$$rwd = 0$$

Con esto concluye el cálculo de la *rwd*, y así se obtiene el segundo parámetro que el *obj_entorno* devuelve al agente.

3.5. Módulo agente

El agente que se implementa para controlar el módulo entorno es el algoritmo de optimización de política próxima (PPO, por sus siglas en inglés) de la biblioteca *Stable Baselines 3*. Este algoritmo PPO utiliza la política perceptron multicapa (MlpPolicy, por sus siglas en inglés). El módulo entorno a controlar es el *obj_ entorno* y se pasa como argumento al algoritmo PPO. Con el agente y sus dos argumentos (la política y el entorno), se tiene todo lo necesario para implementar el sistema de aprendizaje por refuerzo (RL).

Para el entrenamiento, el método *learn* del agente llama internamente a un método *train*, el cual entrena a la política. Para realizar el entrenamiento de la política se implementa en ciclo de búsquedas y episodios internamente en la librería. Este ciclo permite interactuar con el entorno, el agente, recompensas y estados. Durante el proceso de aprendizaje encuentra errores, ajusta la política y realiza otras búsquedas de forma interactiva. El proceso concluye cuando la recompensa se maximiza y se mantiene estable.

Con el modelo entrenado, el siguiente paso es la predicción de *queries* por parte del agente hacia el entorno mediante un bucle *for*. Para llevar a cabo estas predicciones, el agente hace uso del método *predict*. Es importante subrayar, que el entorno va a utilizar una nueva base de datos, el cual el agente desconoce totalmente. De esta forma, el agente va a recibir como argumento el *vector de observaciones* en su método *predict*, procesará este vector con la política aprendida 'MlpPolicy' e inmediatamente emitirá *querys* hacia el entorno. Como resultado final, el sistema buscador personalizado propondrá las conversaciones más cercanas en términos de *tema_ usr* y *complej_ usr*. Las predicciones terminan cuando el bucle *for* concluye.

Capítulo 4

Experimentos y Resultados

Este capítulo se divide en dos secciones principales: la primera aborda la evaluación del rendimiento del agente, mientras que la segunda se enfoca en las predicciones generadas por el mejor modelo evaluado. La sección de evaluación del rendimiento del agente comprende varias subsecciones, que incluyen el diseño de los experimentos, la descripción de los resultados obtenidos y la discusión de estas evaluaciones. Por otro lado, la sección de predicciones con el mejor modelo evaluado consta de subsecciones que detallan los requisitos para llevar a cabo las predicciones, describen los resultados obtenidos y discuten las predicciones realizadas.

4.1. Evaluación de rendimientos

En la fase de diseño de experimentos, se examinan detalladamente diversos parámetros que podrían influir en los resultados de la evaluación del rendimiento del agente. En la sección de descripción de resultados, se llevará a cabo un análisis a detalle de las gráficas derivadas de los experimentos realizados. En la sección de discusión, se profundizará en los aspectos ob-

servados en los resultados experimentales de esta investigación, resaltando los detalles significativos que emergieron durante el proceso de evaluación.

4.1.1. Diseño de experimentos

Para cada uno de los experimentos a llevar a cabo, es esencial tener en cuenta cuatro secciones fundamentales: el agente neuronal, el entorno, la base de datos y los parámetros de configuración. Es importante señalar que los parámetros de configuración propuestos se basaron en numerosas pruebas previamente realizadas. Estos componentes juegan un papel crucial en la planificación y ejecución de los experimentos, ya que influyen en los resultados obtenidos y en la comprensión general del rendimiento del sistema.

4.1.1.1. Experimento 1

Agente Neuronal

La fase inicial de preparación de los experimentos se centró en entrenar al agente neuronal utilizando el algoritmo de aprendizaje por refuerzo. En este y los experimentos subsiguientes, el agente neuronal bajo evaluación es el modelo PPO. La librería para implementar este algoritmo es 'stable_baselines3'.

Los hiperparámetros que se plantean para configurar a PPO son los siguientes:

- *política*: 'MlpPolicy'.
- *n_steps*: 150,000 pasos.

Base de datos

La base de datos utilizada para llevar a cabo este y los experimentos subsiguientes está en formato Excel. Este archivo alberga todas las conversaciones junto con sus atributos descriptivos, como la complejidad de vocabulario.

Entorno

Para poder configurar el entorno, se hace uso de la librería 'GYM' (abreviatura de 'OpenAI Gym').

Los parámetros propuestos para el *obj_entorno* son los siguientes:

- *tema_usr*: Food
- *complej_usr*: 0.2
- *rwd_thd*: 0.5
- *c_thd*: 0.1
- *s_thd*: 0.01
- *action_space*: Tipo multidiscreto con 4 argumentos de tamaño *vocab_tam*. Cada argumento del *action_space* es una palabra en el *query* del agente.
- *observation_space*: Tipo Box de tamaño $vocab\_tam + 2$ (*simi_mediana* y *complej_mediana*).
- *done*: La condición de paro *done* para este experimento 1, va a estar implementado por ventanas. Esta implementación se detalla en el apartado 'diseño de la condición de paro done para el Experimento 1'.

Es importante destacar que el *vector de observaciones* se configura según el espacio de observaciones (etiquetado como *observation_space*). Sin embargo, para el Experimento 1 en el *observation_space* no se incluyeron las ecuaciones 3.1, 3.2 y 3.3. Esto se debe al diseño propio del experimento. Por otro lado, el *query* se ajusta según el espacio de acciones (etiquetado como *action_space*).

Diseño de la condición de paro done para el Experimento 1

Con el objetivo de optimizar el rendimiento del agente, se propone implementar ventanas en la clase Entorno con el objetivo de monitorear el desempeño de la *rwd_acc* durante cada episodio. Estas ventanas tendrán una longitud de 200 pasos y requerirán un porcentaje mínimo de recompensa acumulada (etiquetado como *rwd_min*) del 80 % en cada ventana. Esta iniciativa de diseño para la condición *done* a través de ventanas se planteó de manera alternativa.

Para analizar el comportamiento de las *rwd_acc* obtenidas por el agente durante el entrenamiento, se han definido varios intervalos de tiempo para la evaluación, estos son: 10k, 40K, 80k y 150K pasos, donde el carácter 'K' es equivalente a la unidad mil.

Pseudocódigo y prueba de escritorio para las ventanas

La prueba de escritorio resulta esencial para la validación lógica del algoritmo. Este proceso facilita la depuración y verificación del código, asegurando su funcionamiento adecuado. A continuación, se presenta el pseudocódigo destinado a implementar las ventanas:

Algorithm 1 Pseudocódigo ventanas

```
tam_ventana ← 200
tam_ventana_fija ← tam_ventana
porcent ← 80 %
rwd_min ← (tam_ventana ∗ porcent/100)
rwd_min_fija ← rwd_min
contador ← 0
if contador == tam_ventana and rwd_acc >= rwd_min then
  tam_ventana+ = tam_ventana_fija
  rwd_min = rwd_min_fija + rwd_acc
  done = Falso
else
  if contador == tam_ventana and rwd_acc <rwd_min then
    tam_ventana = tam_ventana_fija
    done = Verdadero
  else
    done = Falso
  end if
end if
```

La tabla 4.1 ilustra el proceso de prueba de escritorio aplicado al algoritmo 1.

Tabla 4.1: Prueba de escritorio del algoritmo 1.

renglón	**contador**	**tam_ventan**	**rwd_acc**	**rwd_min**	**done**
1	0	200	0	160	
2	200	200	160	160	Falso
3		400		320	
4	400	400	320	320	Falso
5		600		480	
6	600	600	400	480	Verdadero
7	0	200	0	160	

Esta prueba comienza en el renglón número 1 con los valores iniciales asignados en el *obj_entorno*. En el renglón número 2 el contador alcanza 200 iteraciones sobre la base de datos y el agente obtiene una *rwd_acc* de 160, lo que satisface la primera condición if. En consecuencia, el tamaño de

ventana aumenta en 200 pasos, llegando a un valor de 400. La *rwd_ min* se actualiza a 320 debido a la suma entre la *rwd_ acc* y la *rwd_ min_ fijo*. La condición *done* se mantiene en falso.

En el renglón número 3, se actualizan los valores del tamaño de ventana y el *rwd_ min*. En el renglón número 4, la primera condición if sigue cumpliéndose, ya que la *rwd_ acc* alcanza un mínimo de 320 cuando el contador llega a 400 iteraciones sobre la base de datos. El *done* permanece en Falso.

En el renglón número 5, se actualiza el tamaño de la ventana, aumentando en 200 unidades y alcanzando un valor de 600. La *rwd_ min* se incrementa a 480 debido a la suma entre la *rwd_ acc* y la *rwd_ min_ fijo*. En el renglón número 6, ya no se cumple la primer condición if, dado que la *rwd_ acc* alcanza el valor de 400 y el *rwd_ min* esperado era 480. Por lo tanto, la condición de paro *done* se vuelve verdadero y el episodio concluye. En el renglón 7, se reinician los valores iniciales del algoritmo y se inicia otro episodio.

La introducción de estas ventanas y el uso del parámetro *rwd_ min* en la medición, permitirá al agente evitar la activación prematura de la condición de paro *done*, lo que a su vez conducirá a obtener una *rwd_ acc* más alta durante la extensión de cada episodio. Es importante resaltar que la duración de cada episodio se ajustaba en relación a cuántas ventanas se podrían alcanzar con el parámetro *rwd_ min*.

4.1.1.2. Experimento 2

En este experimento 2, se toma como base el diseño de la clase Entorno visto en la sección (3.4), salvo algunos ajustes que se mencionan a continuación:

Agente Neuronal

En el Experimento 2, se mantiene el mismo algoritmo de aprendizaje por refuerzo que en el Experimento 1, es decir, el modelo PPO.

Los hiperparámetros que se plantean para configurar este modelo son los siguientes:

- *política*: 'MlpPolicy'.
- *n_steps*: 800,000 pasos.

Base de datos

La base de datos se mantiene igual, sin ningún cambio.

Entorno

Con el objetivo de maximizar la *rwd_acc* en cada episodio, se plantea la estrategia de utilizar los datos más frecuentes presentes en la base de datos. En este contexto, cuando se mencionan datos más frecuentes se hace referencia a los temas más preponderantes en las conversaciones, así como a las complejidades más representativas. Esta estrategia busca enfocar al agente en situaciones y contextos que son más comunes y que reflejan la distribución real de temas y complejidades en las conversaciones disponibles en la base de datos.

Por consiguiente, se ha realizado una modificación inicial en la clase Entorno al introducir un nuevo enfoque para la selección del *tema_usr*. En lugar de ingresar manualmente el *tema_usr*, ahora se obtiene utilizando un proceso de selección de temas preponderantes presentes en la base de datos. Este proceso ha sido respaldado por un algoritmo previamente validado en el campo [40]. La figura 4.1 ilustra el procedimiento para la obtención de

los temas más preponderantes de la base de datos:

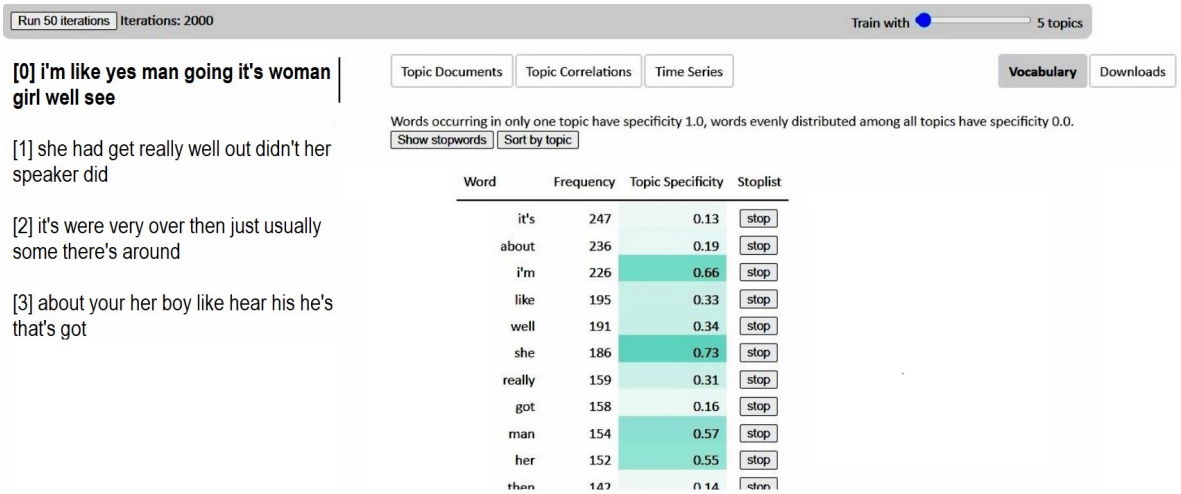


Figura 4.1: Temas preponderantes de la base de datos

En este experimento, se plantea la elección de los primeros temas de usuario que el algoritmo generó como preponderantes. Para alcanzar estos temas preponderantes, el algoritmo requirió de un archivo que contiene las denominadas 'stop words'. Estas 'stop words' son palabras comunes que se utilizan con alta frecuencia en un lenguaje, pero que carecen de un significado específico o semántico.

El segundo ajuste realizado en la clase Entorno implica la modificación de la variable *complej_usr*. Para lograr esto, se implementa un análisis de datos utilizando un diagrama de caja y bigotes sobre el campo de 'complejidades' en la base de datos. La figura 4.2 muestra el diagrama de bigotes utilizado en este proceso:

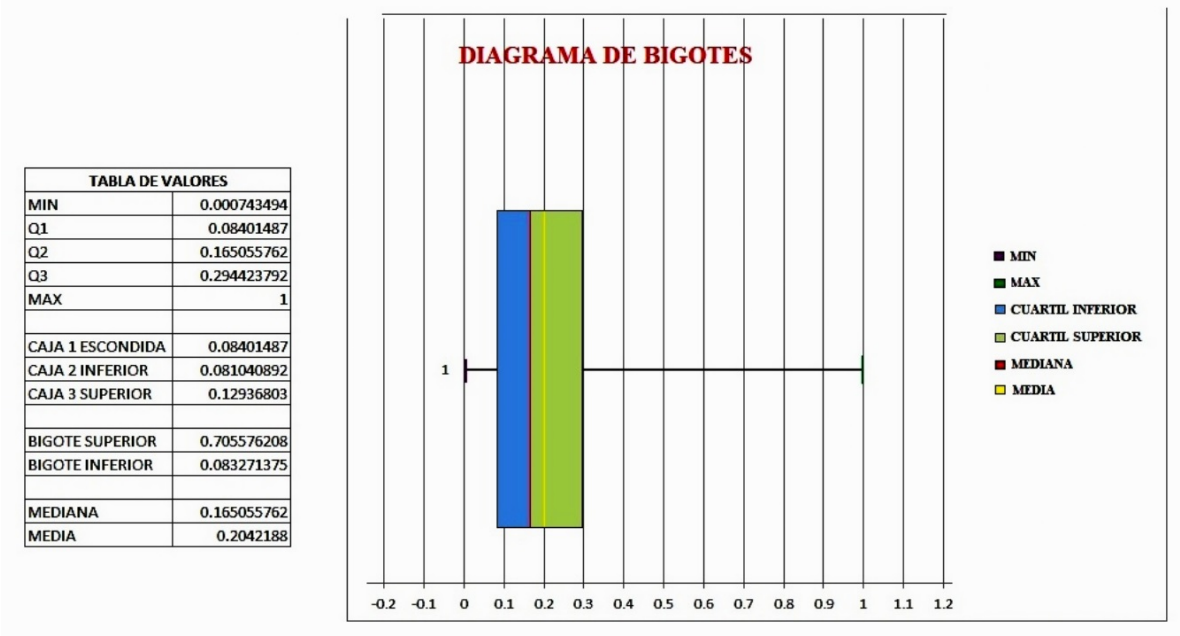


Figura 4.2: Diagrama de cajas y bigotes

Como se puede observar en la figura 4.2 el valor de la *complej_mediana* (abreviado como *CM*) es de 0.1650 y se sitúa en la parte inferior del cuartil. Esta gráfica ilustra que los valores de complejidad en las conversaciones de la base de datos están mayoritariamente en el rango entre el nivel medio y el nivel básico. Con esta información, se plantea la realización de experimentos utilizando tanto el valor de la *CM* como el valor de complejidad del cuartil inferior (abreviado como *CQ*) como parámetros para la *complej_usr*.

En resumen, los parámetros propuestos para el objeto *obj_entorno* son los siguientes::

- *tema_usr*: Los tres temas más preponderantes.
- *complej_usr*: Las complejidades *CM* y *CQ*.
- *rwd_thd*: 0.5
- *c_thd*: 0.1

- *s_thd*: 0.01
- *action_space*: Tipo multidiscreto, con 2 y 4 argumentos de tamaño *vocab_tam*. La combinación de la longitud del *query* con el *tema_usr* y la *complej_usr* se detalla en el apartado que lleva por nombre 'Tabla de combinación del Experimento 2'.
- *observation_space*: Tipo Box de tamaño *vocab_tam* + 2 (*simi_mediana* y *complej_mediana*).
- *done*: La condición de paro *done* para este experimento 2, va a estar implementado con referencia a los umbrales de la complejidad y la similitud. Esta implementación se detalla en el apartado 'Diseño de la condición de paro done para el Experimento 2'.

Es importante destacar que en este contexto, tanto el *action_space* como el *observation_space* se mantienen configurados de manera similar a como se establecieron en el Experimento 1.

Diseño de la condición de paro done para el Experimento 2

El tercer ajuste que se realiza en la clase Entorno es la configuración de la condición de paro *done*. Esta condición de paro se establece con referencia a los umbrales *c_thd* y *s_thd*. En el Experimento 2, el objetivo es mantener un rango de complejidades cercanas a la *complej_usr*, por lo que se definen una complejidad inferior (etiquetada como *c_inf*) y una complejidad superior (etiquetada como *c_sup*). La *c_inf* se calcula como la diferencia entre la *complej_usr* y *c_thd*, mientras la *c_sup* se calcula como la suma de la *complej_usr* y *c_thd*.

Además, es importante garantizar que la *simi_mediana* (similitud mediana) no supere el umbral *s_thd*. Debido a esto, la condición *done* se restringe en función de la *complej_usr* y la similitud con el *tema_usr*. En

la eq. 4.1 se expresan las condiciones que deben cumplirse para activar el *done*:

$$c\_inf = complej\_usr - c\_thd$$

$$c\_sup = complej\_usr + c\_thd$$

$$done = \text{bool}[(CM => c\_sup) \vee (CM <= c\_inf) \vee (simi\_mediana <= s\_thd)] \quad (4.1)$$

La finalidad de la Eq. 4.1 es posicionar la *complej_mediana* (CM) de manera equidistante entre los umbrales de las complejidades, acercándola lo máximo posible a la *complej_usr*. Por otro lado, la *simi_mediana* debe ubicarse por encima del umbral de similitud, es decir, su valor debe tender hacia 1.

Tabla de combinación del Experimento 2

El cuarto ajuste efectuado a la clase Entorno se centra en la configuración del *action_space*. Los cuatro argumentos contenidos en el *action_space*, utilizados para generar *query*, serán combinados con las *complej_usr* propuestas en este Experimento 2, junto al tamaño de los tres temas preponderantes extraídos de la base de datos. Los detalles de estas combinaciones se presentan en la Tabla 4.2, la cual muestra la matriz de combinaciones para la realización del Experimento 2.

Tabla 4.2: Matriz de combinación *query* y *tema_usr*.

	CM=0.165	**CQ=0.084**
Tema preponderante 1	Q2_T3, Q2_TF, Q4_T3, Q4_TF	Q2_T3, Q2_TF, Q4_T3, Q4_TF
Tema preponderante 2	Q2_T3, Q2_TF, Q4_T3, Q4_TF	Q2_T3, Q2_TF, Q4_T3, Q4_TF
Tema preponderante 3	Q2_T3, Q2_TF, Q4_T3, Q4_TF	Q2_T3, Q2_TF, Q4_T3, Q4_TF

donde:

Q2: son dos argumentos del *action_ space*.
Q4: son cuatro argumentos del *action_ space*.
T3: son tres palabras del *tema_ usr*.
TF: son once palabras del *tema_ usr*.

El parámetro de salida a evaluar del *obj_ entorno* es la *rwd_ acc* durante el tiempo de cada episodio. El intervalo de confianza para el promedio de la recompensa acumulada será del 95 por ciento. El intervalo de confianza es un intervalo de valores que proporcionan una estimación de la media con cierto grado de certeza.

4.1.1.3. Experimento 3

En el Experimento 3, se partirá del sistema implementado en el Experimento 2 como base. A continuación, se describen los ajustes realizados en el agente neuronal PPO, la base de datos y el entorno:

Agente Neuronal

En el marco del Experimento 3, se establecen los siguientes hiperparámetros para la configuración del modelo PPO:

- *política*: 'MlpPolicy'.
- *Número de capas ocultas*: 2
- *Número de nodos*: 64
- *n_ steps*: 800,000 pasos.

El primer ajuste que se realiza en el Experimento 3 es la configuración de hiperparámetros al modelo PPO. Estos se realizan con el objetivo de

explorar y evaluar cómo esta nueva configuración impacta en el rendimiento y la capacidad de aprendizaje del agente.

Base de datos

El segundo ajuste a realizar en este Experimento 3 implica ampliar el contenido de la base de datos, logrando un aumento significativo en el número de conversaciones de distintos niveles: básico, intermedio y avanzado. A diferencia del Experimento 2, donde se utilizaron 200 conversaciones, en esta fase se emplearán 2,000 nuevas conversaciones. Este aumento en la cantidad de datos permitirá una mayor diversidad y cobertura en los niveles de dificultad, enriqueciendo de manera considerable el conjunto de entrenamiento para el agente neuronal PPO.

Entorno

Con la integración de la nueva base de datos, se avanza hacia el tercer ajuste, que se centra en la determinación del *tema_usr*. Para llevar a cabo este proceso, se retoma la aplicación del algoritmo 'JDA' previamente utilizado en el Experimento 2 para la selección de temas preponderantes. La Figura 4.3 se presenta de manera gráfica el proceso de obtención de los temas más destacados en la nueva base de datos.

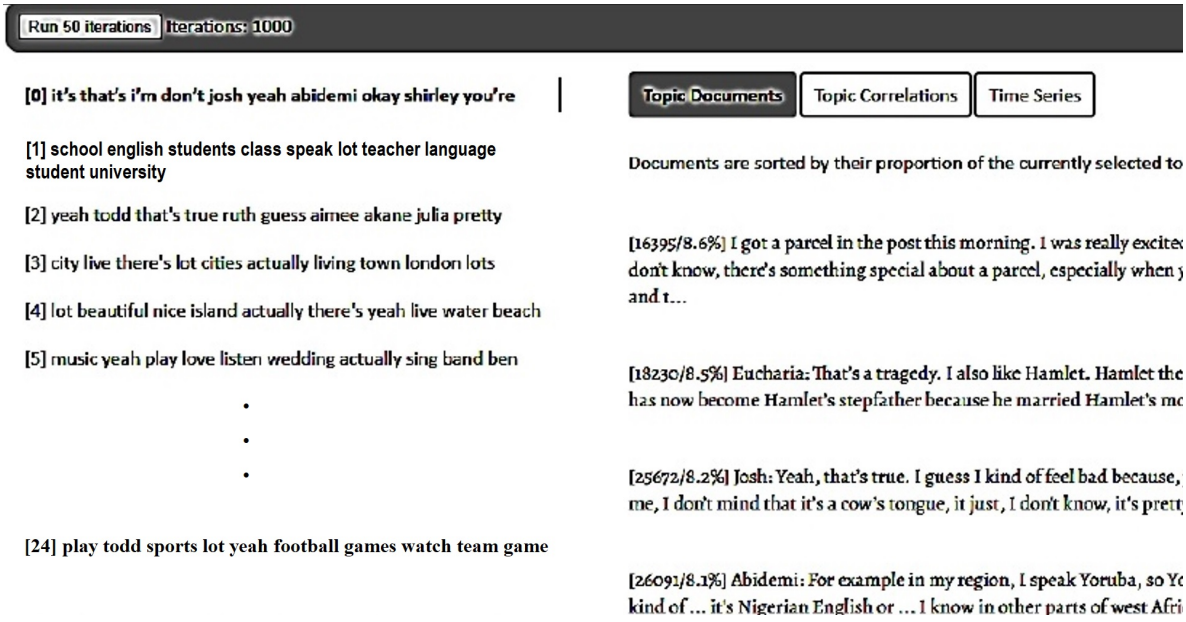


Figura 4.3: Temas preponderantes con la nueva base de datos

En el contexto de este Experimento 3, se propone la elección de únicamente dos temas entre los identificados como preponderantes, los cuales serán los que ocupan los extremos de la lista de resultados. Tras aplicar el algoritmo, se lograron identificar un total de 24 temas de gran relevancia. Como resultado de esta identificación, se tomó la decisión de designar al tema 1 como el más preponderante y al tema 24 como el menos preponderante, ambos como propuestas de *tema_usr*.

Una vez que la nueva base de datos ha sido integrada, se procede a realizar el cuarto ajuste, que implica calcular los valores correspondientes de la *complej_usr*. Para llevar a cabo esta tarea, se realiza nuevamente el análisis mediante diagrama de caja y bigotes en el atributo 'complejidades' en la base de datos. En la figura 4.4 se muestra el diagrama de bigotes correspondiente a este análisis.

TABLA DE VALORES	
MIN	0.00000
Q1	0.06255012
Q2	0.115076183
Q3	0.170809944
MAX	1.00000
CAJA 1 ESCONDIDA	0.06255012
CAJA 2 INFERIOR	0.052526063
CAJA 3 SUPERIOR	0.055733761
BIGOTE SUPERIOR	0.829190056
BIGOTE INFERIOR	0.06255012
MEDIANA	0.11508
MEDIA	0.12499

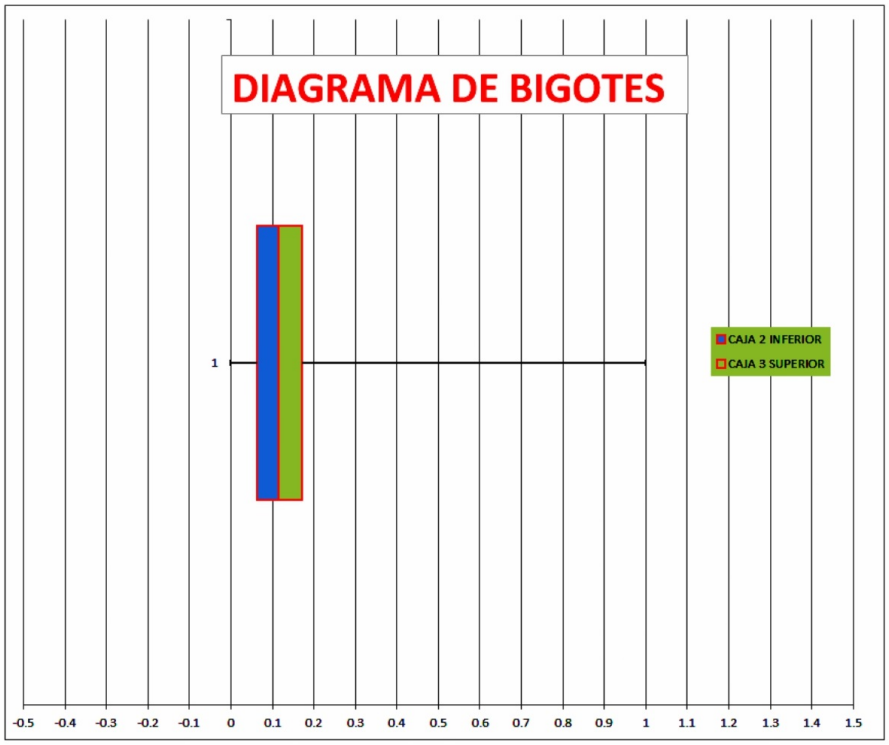


Figura 4.4: Diagrama de cajas y bigotes

En la figura 4.4, se puede apreciar que el valor de la mediana es 0.115 y se encuentra inclinado hacia el cuartil inferior. Esta gráfica evidencia que los valores de las complejidades en la base de datos se encuentran mayormente en un rango entre el nivel medio y básico. Teniendo en cuenta esta información, se propone que la *complej_usr* tome los valores CM y CQ para llevar llevar a cabo el Experimento 3.

En resumen, los parámetros propuestos para el *obj_entorno* son los siguientes::

- *tema_usr*: es el tema más y menos preponderante.
- *complej_usr*: son las complejidades *CM* y *CQ*.
- *rwd_thd*: 0.5

- *c_thd*: 0.1
- *s_thd*: 0.01
- *action_space*: es de tipo multidiscreto, con 1 y 5 argumentos de tamaño *vocab_tam*, donde cada argumento va a tener sus propias combinaciones. Estos ajustes se detallan en el apartado 'Tabla de combinación del Experimento 3'.
- *observation_space*: es de tipo Box de tamaño *vocab_tam* + 4 (*simi_mediana*, *complej_mediana*, Δ*_simi* y Δ*_complej*).
- *done*: la condición de paro *done* para este experimento 3 va a estar implementado con referencia al sistema CartPole. Esta implementación se detalla en el apartado que lleva por nombre 'Diseño de la condición de paro *done* para el Experimento 3'.

Es importante destacar que en este contexto, el Experimento 3 contempla un quinto ajuste que involucra la implementación del *vector de observaciones* basado en los procesos de decisión de Markov (MDP). Esto implica que las ecuaciones 3.1, 3.2 y 3.3 son aplicadas en el *vector de observaciones*, junto con su correspondiente configuración en el *observation_space*.

Diseño de la condición de paro *done* para el Experimento 3

El sexto ajuste que se realiza al Experimento 3 es la configuración de la condición de paro *done*. Esta condición de paro se establece con referencia al sistema CartPole visto en la sección 2.3.3. De acuerdo con la Eq. (2.11) la condición de paro *done* para el Experimento 3 queda de la siguiente manera:

$$done = \text{bool}[((complej\_usr \text{ - } complej\_mediana) \geq c\_thd) \vee (simi\_mediana \leq s\_thd)] \quad (4.2)$$

La Eq. 4.2 establece un umbral para la diferencia de complejidades entre la que ingresa el usuario y la que regresa el sistema buscador, esta diferencia no debe estar por encima del umbral. Esto quiere decir que las complejidades deben estar lo más cercano posible en el mejor de los casos.

Por otro lado, en términos de *similitud*, la Eq. 4.2 dice que la semejanza que exista entre el tema de usuario y las conversaciones que regrese el buscador, no debe estar por debajo de su umbral. Esto quiere decir, que la semejanza tienda al valor 1 en el mejor de los casos.

Adicionalmente, se agrega otro ajuste a la condición de paro *done*, con el fin de garantizar todos los procesos de aprendizajes que lleve a cabo el agente. Mediante la realización de varios experimentos se establece un paro forzoso después de que *n_steps* cumpla su límite de pasos de entrenamiento. Esto asegura que el último episodio durante el entrenamiento sea considerado por el algoritmo PPO. De esta manera, se obtendrá un registro amplio de los diferentes niveles de aprendizaje que alcance el agente durante su entrenamiento.

Tabla de combinación del Experimento 3

Se realiza un séptimo ajuste en el Experimento 3, con la implementación de 1 y 5 argumentos en la configuración del *action_space*. Con la introducción de la nueva longitud del *query*, este se combinará con las *complej_usr* y los temas preponderantes propuestos. En la Tabla 4.3 se presenta la matriz de combinaciones utilizada para llevar a cabo el Experimento 3.

Tabla 4.3: Matriz de combinación *query* y *tema_usr* con nueva base de datos

K	**Tema**	**CM=0.115**	**CQ=0.062**
1	Tmas	(1a) Q5 y T11, (1b) Q1 y T3	(5a) Q5 y T11, (5b) Q1 y T3
	Tmenos	(2a) Q5 y T11, (2b) Q1 y T3	(6a) Q5 y T11, (6b) Q1 y T3
5	Tmas	(3a) Q5 y T11, (3b) Q1 y T3	(7a) Q5 y T11, (7b) Q1 y T3
	Tmenos	(4a) Q5 y T11, (4b) Q1 y T3	(8a) Q5 y T11, (8b) Q1 y T3

donde:
K=1: es un resultado de búsqueda en la base de datos.
K=5: son cinco resultados de búsqueda en la base de datos.
Tmas: es el tema más preponderante en la base de datos.
Tmenos: es el tema menos preponderante en la base de datos.
CM: es la complejidad mediana en el diagrama de cajas y bigotes.
CQ: es la complejidad del cuartil inferior en el diagrama de cajas y bigotes
Q1: es un argumento del *action_space*.
Q5: son cinco argumentos del *action_space*.
T3: son tres palabras del *tema_usr*.
T11: son once palabras del *tema_usr*.

4.1.2. Descripción de los resultados

4.1.2.1. Experimento 1

En la Figura 4.5, se presenta la gráfica que ilustra la evolución de la *rwd_acc* obtenida por el agente en relación al tema de usuario 'Food' y una *complej_usr* establecida en 0.2. Esta representación abarca un intervalo de 10,000 pasos y brinda una visión detallada del rendimiento del agente en este intervalo especifico.

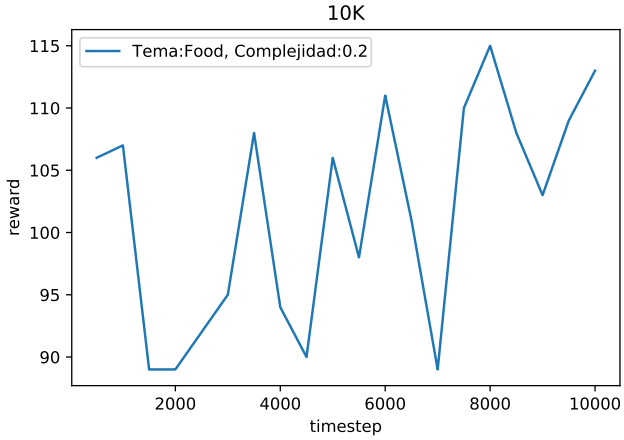


Figura 4.5: Gráfica de rwd_acc en 10K pasos

En la gráfica que abarca 10,000 pasos, se puede notar que el agente se encuentra en una fase inicial de aprendizaje en términos de cómo realizar consultas efectivas. Esto se refleja en la *rwd_ acc*, la cual muestra un valor bastante bajo y apenas presenta un incremento considerable.

En la Figura 4.6 se representa la gráfica de la *rwd_ acc* alcanzada por el agente al utilizar el tema de usuario 'Food' y una *complej_ usr* de 0.2, a lo largo de 40,000 pasos.

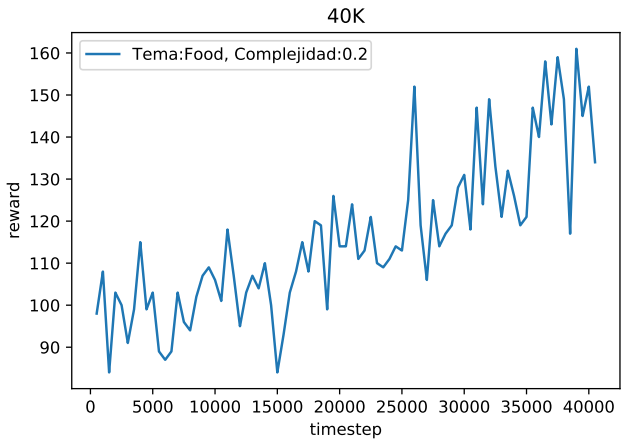


Figura 4.6: Gráfica de rwd_acc en 40K pasos

En esta gráfica correspondiente a 40,000 pasos, se puede observar que el agente está adquiriendo una mayor habilidad en la realización de consultas. Esto se refleja en el ligero incremento de la *rwd_ acc* en su escala.

La figura 4.7 presenta la gráfica de la *rwd_ acc* obtenida por el agente con el tema de usuario 'Food' y una *complej_ usr* de 0.2, en un período de 80,000 pasos.

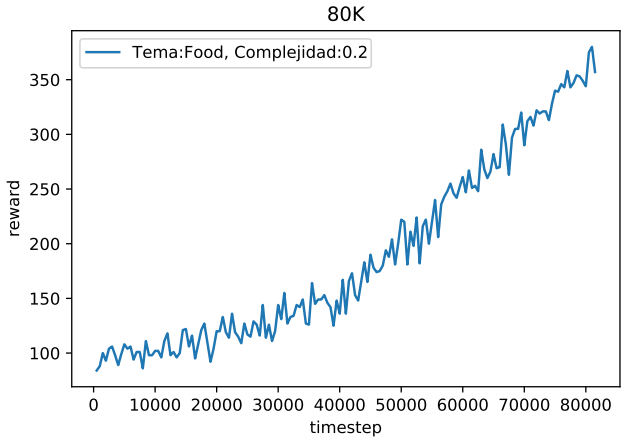


Figura 4.7: Gráfica de rwd_acc en 80K pasos

La gráfica correspondiente a 80,000 pasos (figura 4.7) revela que el agente está adquiriendo una mayor destreza en la realización de consultas. Esto se refleja en el significativo aumento de la *rwd_ acc* en la escala de valores, aunque aún no ha alcanzado una estabilización completa.

La figura 4.8 presenta la gráfica de la *rwd_ acc* obtenida por el agente con el tema de usuario 'Food' y una *complej_ usr* de 0.2, en un intervalo de 150,000 pasos.

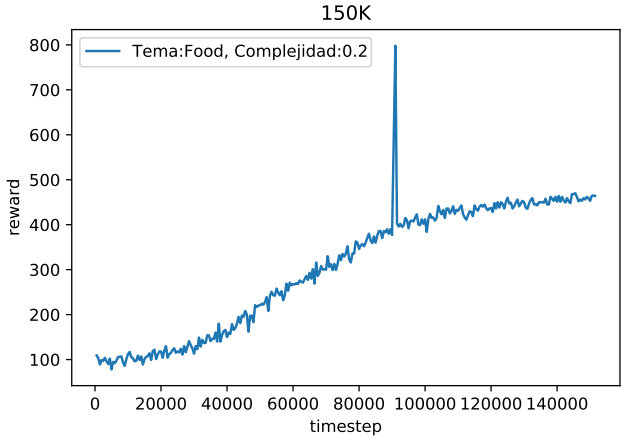


Figura 4.8: Gráfica de rwd_acc en 150K pasos

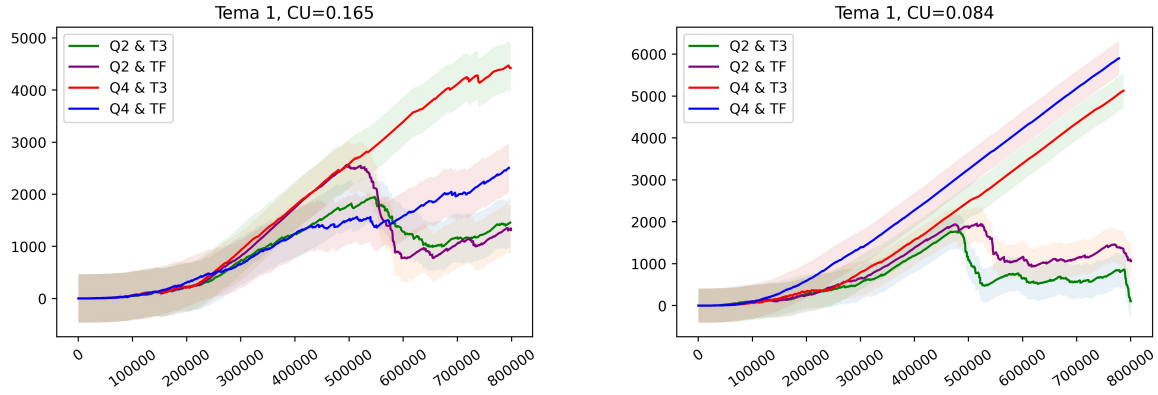
En la gráfica correspondiente a 150,000 pasos (figura 4.8), se puede apreciar que el agente ha adquirido la habilidad de generar consultas más cercanas a las preferencias del usuario. Este avance se refleja en el aumento moderado de la *rwd_ acc* igual a 500 y su tendencia hacia la estabilización.

Al finalizar Experimento 1, se pudo observar a través de las gráficas que la *rwd_ acc* tiende a estabilizarse después de alcanzar los 150,000 pasos. Sin embargo, es importante destacar que la activación del 'done' depende exclusivamente de la condición de *rwd_ min* en las ventanas establecidas. En este mismo contexto, es preciso mencionar que en este experimento, la implementación del módulo entorno no siguió el diseño del CartPole de la librería OpenAI Gym. Esto implica que no se basó en la analogía de los parámetros críticos que deben activar la condición de paro *done*, como se establece en el CartPole.

4.1.2.2. Experimento 2

En la figura 4.9 se presentan dos gráficas que muestran los resultados obtenidos a partir de las dos combinaciones entre el tema preponderante 1

y los valores de las dos complejidades. Cada combinación refleja la variación en el tamaño del *query* y el tamaño del *tema_usr*.



(a) (b)

Figura 4.9: Tema preponderante 1.

En la figura 4.9a, se puede apreciar que para el conjunto de combinaciones Q2_T3 y Q2_TF, las *rwd_mean* alcanzan su valor máximo alrededor del paso 550,000, llegando a aproximadamente 2000, después empiezan a disminuir hasta llegar a alrededor de 700 cuando alcanzan el paso 690,000. Posteriormente, vuelven a aumentar. Los intervalos de confianza para las combinaciones Q2_T3 y Q2_TF se superponen durante todo el período.

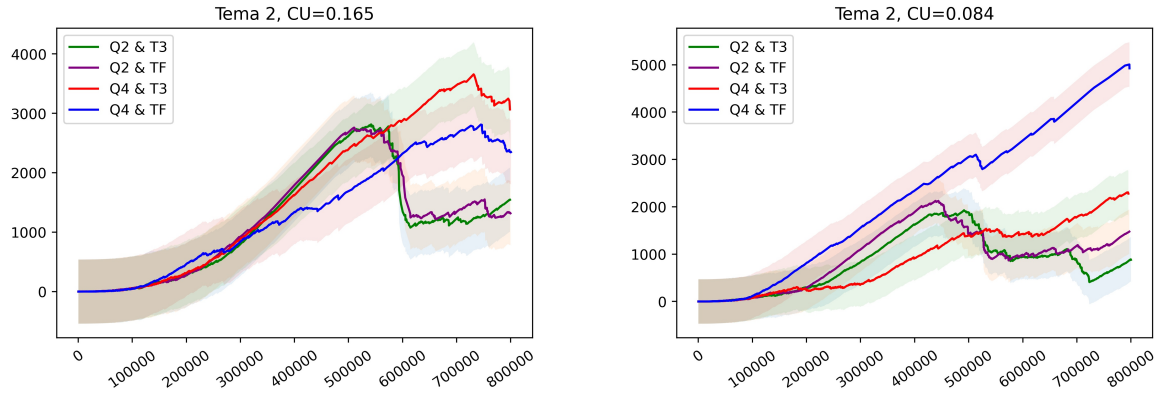
Por otro lado, las combinaciones Q4_TF y Q4_T3 muestran tendencias ascendentes en las *rwd_mean*. Mientras Q4_TF alcanza su máximo alrededor de 2500, Q4_T3 tiene un pico de alrededor de 4500. Los intervalos de confianza para las combinaciones Q4_TF y Q4_T3 tienden a separarse. Ninguna de las combinaciones muestra estabilidad en sus valores.

En la figura 4.9b, se puede observar que en el conjunto de combinaciones Q2_T3 y Q2_TF, las *rwd_mean* alcanzan su valor máximo alrededor del paso 500,000, llegando a alrededor de 1900. Posteriormente, empiezan a disminuir hasta llegar a alrededor de 900 cuando alcanzan el paso 600,000,

y luego vuelven a caer gradualmente, tendiendo a cero. Los intervalos de confianza de las combinaciones Q2_T3 y Q2_TF se superponen a lo largo de todo el período.

Por otro lado, en el conjunto de combinaciones Q4_TF y Q4_T3, se observa una tendencia ascendente en las *rwd_mean*. Mientras Q4_TF alcanza su máximo alrededor de 5800, Q4_T3 tiene un pico de alrededor de 5000. Los intervalos de confianza de Q4_TF y Q4_T3 se superponen durante todo el período. En ninguna de las gráficas se observa estabilidad en los valores.

Continuando con la siguiente configuración, la figura 4.10 muestra dos gráficas que surgen de las dos combinaciones entre el segundo tema preponderante y los valores de las dos complejidades (CM y CQ). Cada una de estas combinaciones implica variaciones en el tamaño del *query* y el tamaño del *tema_usr*.



(a) (b)

Figura 4.10: Tema preponderante 2.

En la figura 4.10a, se puede apreciar que en el conjunto de combinaciones Q2_T3 y Q2_TF, las *rwd_mean* alcanzan su valor máximo alrededor del paso 550,000, llegando a aproximadamente 3000. Luego, empiezan a disminuir hasta alrededor de 1000 en el paso 600,000, manteniéndose rela-

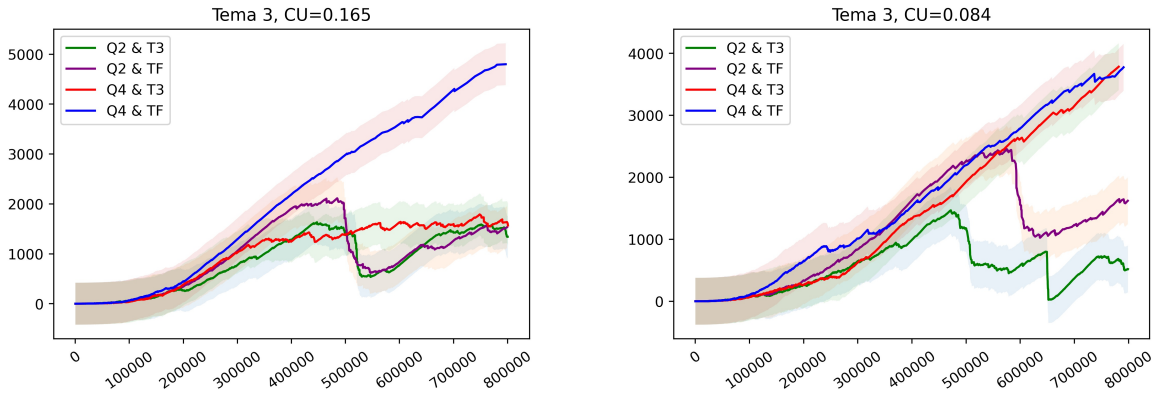
tivamente estables posteriormente. Los intervalos de confianza para Q2_T3 y Q2_TF se superponen a lo largo de todo el proceso de entrenamiento.

Por otro lado, las otras dos combinaciones, Q4_TF y Q4_T3, presentan una tendencia ascendente en las *rwd_mean*. La combinación Q4_TF llega a un máximo de alrededor de 2700, mientras que Q4_T3 alcanza un pico de 3800. Los intervalos de confianza de estos dos conjuntos también se superponen durante el entrenamiento. En general, ninguna de las gráficas muestra estabilidad a lo largo del proceso.

En la figura 4.10b, se puede observar que en el conjunto de combinaciones Q2_T3 y Q2_TF, las *rwd_mean* alcanzan su valor máximo alrededor del paso 450,000, llegando a aproximadamente 2000. Posteriormente, empiezan a disminuir hasta llegar a alrededor de 900 cuando se alcanza el paso 600,000, para luego mostrar un incremento y tender hacia 1000. En lo que respecta al conjunto Q2_T3 y Q2_TF, sus intervalos de confianza tienden a superponerse a lo largo del entrenamiento.

En cuanto al conjunto de combinaciones Q4_TF y Q4_T3, las *rwd_mean* muestran una tendencia ascendente. Mientras Q4_TF alcanza un valor máximo de alrededor de 5000, Q4_T3 llega a aproximadamente 2000. Los intervalos de confianza para el conjunto Q4_TF y Q4_T3 tienden a separarse durante el entrenamiento. En resumen, ninguna de las gráficas muestra estabilidad en el proceso de entrenamiento.

Continuando con las dos últimas combinaciones, en la figura 4.11 se presentan dos gráficas que muestran los resultados obtenidos a partir de las dos combinaciones entre el tema preponderante 3 y los valores de las dos complejidades (CM y CQ). Cada una de estas combinaciones varía en función del tamaño del *query* y del tamaño del *tema_usr*.



(a) (b)

Figura 4.11: Tema preponderante 3.

En la figura 4.11a se puede observar que en el conjunto de combinaciones Q2_T3 y Q2_TF, las *rwd_mean* alcanzan un valor máximo alrededor del paso 500,000, luego comienzan a disminuir hasta llegar a un valor cercano a 500 alrededor del paso 550,000, y luego vuelven a aumentar llegando a alrededor de 1500. Los intervalos de confianza para la combinación Q2_T3 y Q2_TF se superponen durante todo el período.

En cuanto al otro conjunto de combinaciones, Q4_TF y Q4_T3, se observa una tendencia ascendente en las *rwd_mean*. Mientras Q4_TF alcanza un máximo de alrededor de 5000, Q4_T3 tiene un máximo de 1500. A medida que avanza el período, los intervalos de confianza para Q4_TF y Q4_T3 se van separando. Además, el intervalo de confianza de Q4_T3 tiende a superponerse con los intervalos de confianza de Q2_T3 y Q2_TF. En esta gráfica, la combinación que parece mostrar una mayor estabilidad es Q4_T3.

En la figura 4.11b, se puede observar que en la combinación Q2_TF, la *rwd_mean* alcanza su valor máximo, alrededor de 2500, en el paso 580,000. Sin embargo, a partir del paso 630,000, comienza a disminuir y alcanza un valor cercano a 1000. Luego, experimenta un aumento, llegando a alrededor

de 1500. En esta combinación, el intervalo de confianza de Q2_TF tiende a separarse de las otras combinaciones. Por otro lado, en la combinación Q2_T3, la *rwd_mean* alcanza su punto máximo, aproximadamente 1300, en el paso 480,000. No obstante, después de ese punto, la gráfica se vuelve muy inestable y muestra una tendencia descendente. Además, el intervalo de confianza de Q2_T3 tiende a separarse de las otras combinaciones en varios puntos del entrenamiento.

En lo que respecta al conjunto de combinaciones Q4_TF y Q4_T3, ambas muestran una tendencia al alza en las *rwd_mean*, llegando ambas a un valor máximo de alrededor de 4000. Aunque los intervalos de confianza de Q4_TF y Q4_T3 se superponen en todo el período de entrenamiento, ninguna de las gráficas presenta una clara estabilidad en sus resultados.

Al concluir el Experimento 2, se realizaron análisis importantes de los resultados obtenidos, lo cual permitió extraer importantes conclusiones. A continuación, se enumeran cada una de estas conclusiones:

1. La primera combinación, denominada Q2_T3, involucra dos argumentos en el *query* del agente. El *tema_usr* seleccionado consistió en tres palabras relacionadas con el tópico. Esta configuración se aplicó a los tres tópicos preponderantes y a las dos complejidades propuestas. Sin embargo, estas combinaciones de parámetros arrojaron las *rwd_mean* más bajas en comparación con todas las demás combinaciones realizadas. La *rwd_mean* máxima alcanzó aproximadamente los 2800.

2. La siguiente combinación, designada como Q2_TF, también implicó el uso de dos argumentos en el *query* del agente. Sin embargo, en esta configuración, el *tema_usr* consistió en el tópico completo. Se aplicó esta combinación a los tres tópicos preponderantes y a las dos complejidades propuestas. Al final, estas variaciones de parámetros

no resultaron en una mejora significativa en las *rwd_ mean*. De hecho, presentaron un comportamiento bastante similar al de la combinación Q2_T3, con un *rwd_ mean* máximo en torno a 2800.

3. La tercera configuración adoptó la etiqueta Q4_T3. En esta variante, el *query* del agente se extendió de dos a cuatro argumentos. En este caso, el *tema_ usr* consistió en tres palabras del tópico. Se aplicaron estas modificaciones a los tres tópicos preponderantes, junto con las dos complejidades propuestas. Los resultados de estas combinaciones de parámetros generaron mejoras sustanciales. De hecho, las *rwd_ mean* alcanzaron un valor máximo en torno a 5,000. Además de esta mejora en los resultados, la configuración Q4_T3 también mostró una tendencia mayor a estabilizarse en comparación con las otras combinaciones.

4. La cuarta combinación correspondió a Q4_TF. En esta configuración, el *query* del agente mantuvo sus cuatro argumentos, y el *tema_ usr* consistió en el tópico completo. Se llevaron a cabo experimentos utilizando los tres tópicos preponderantes y las dos complejidades propuestas. Estas combinaciones de parámetros generaron las *rwd_ mean* más altas entre todas las realizadas, con un valor máximo cercano a 6,000. Sin embargo, nunca alcanzo la estabilidad.

4.1.2.3. Experimento 3

En la Figura 4.12 se presentan cuatro gráficas que corresponden a las combinaciones 5 y 7 detalladas en la Tabla 4.3. Las gráficas del lado derecho representan la combinación 7, mientras que las del lado izquierdo corresponden a la combinación 5. Estas combinaciones comparten tres valores en común. En primer lugar, el *tema_ usr* es igual a *Tmas*. En segundo

lugar, la *complej_ usr* se establece en *CQ*. El tercer parámetro consiste en dos subcombinaciones: la primera implica un tamaño de *query* de cinco y un tamaño de *tema_ usr* de once, mientras que la segunda subcombinación consta de un tamaño de *query* de uno y un tamaño de *tema_ usr* de tres.

La diferencia clave entre las gráficas del lado derecho e izquierdo radica en el tamaño del resultado de búsqueda, denominado K. En las gráficas del lado izquierdo (combinación 5), el valor de K es igual a uno, mientras que en las gráficas del lado derecho (combinación 7), el valor de K se establece en cinco.

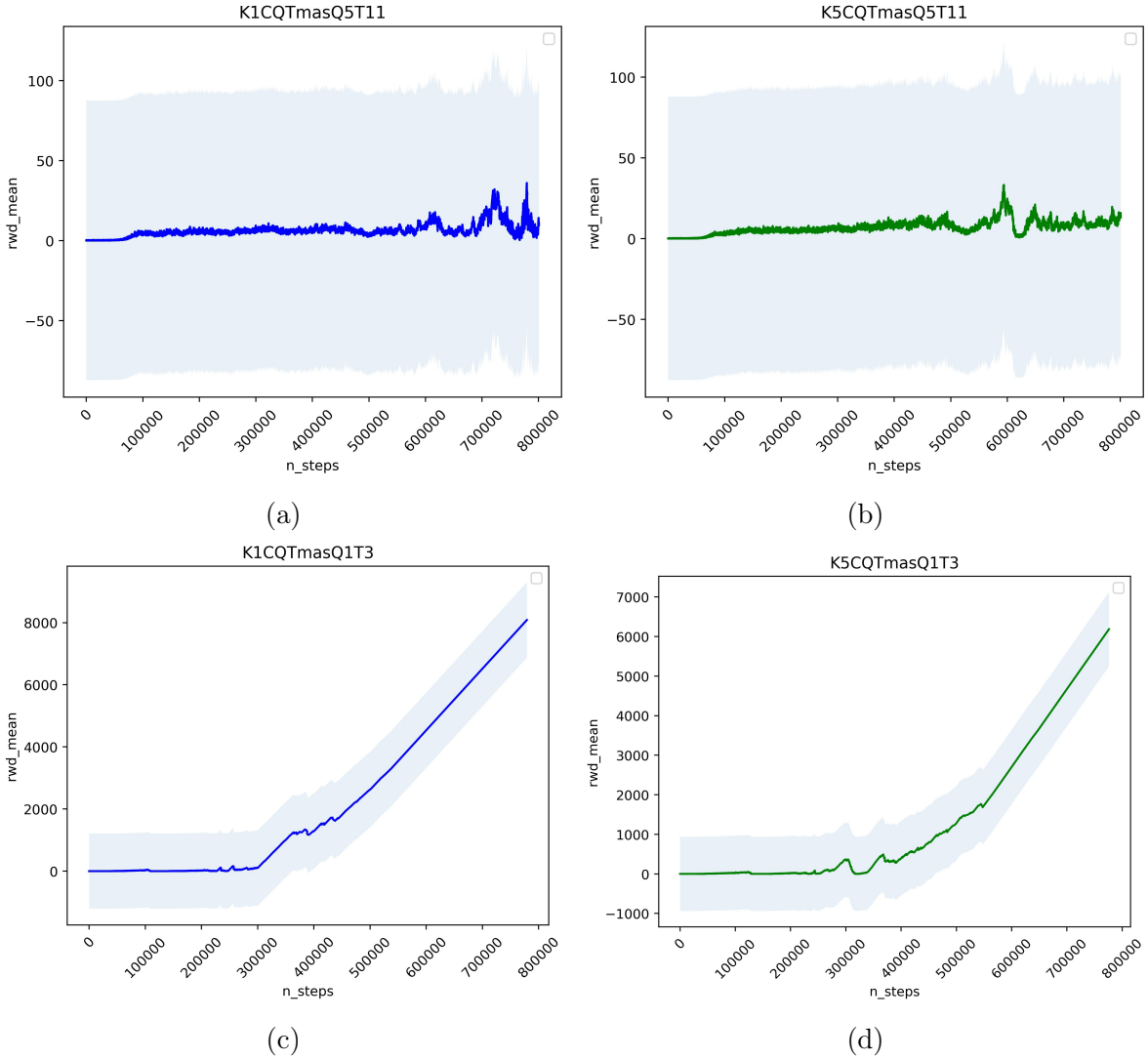


Figura 4.12: Tema más preponderante y complejidad del cuartil inferior.

Las figuras 4.12a y 4.12b representan las combinaciones 5a y 7a de la tabla 4.3, respectivamente. A lo largo del entrenamiento de 800,000 pasos, ambas combinaciones alcanzaron una *rwd_ mean* máxima aproximada de 30. Además, a pesar de tener valores bajos de *rwd_ mean*, tanto las combinaciones 5a como 7a demostraron una notable inestabilidad durante el proceso de entrenamiento.

Por otro lado, las figuras 4.12c y 4.12d corresponden a las combinaciones 5b y 7b de la tabla 4.3, respectivamente. Estas gráficas revelan un aumento significativo en la *rwd_ mean* a medida que avanzan, llegando a alcanzar valores máximos de 8,000 y 6,000 respectivamente al finalizar el entrenamiento. Además de presentar *rwd_ mean* más altos, las combinaciones 5b y 7b exhibieron una mayor estabilidad ascendente en sus curvas de aprendizaje.

La figura 4.13 presenta cuatro gráficas, correspondientes a las combinaciones 6 y 8 de la tabla 4.3. En específico, las gráficas del lado derecho se asocian con la combinación 8, mientras que los gráficos del lado izquierdo están relacionados con la combinación 6. Estas dos combinaciones comparten tres valores clave. El primero de ellos es el *tema_ usr*, que es designado como "Tmenos". El segundo valor compartido es la *complej_ usr*, establecida como CQ. El tercer conjunto de parámetros involucra dos combinaciones distintas. La primera combina un tamaño de *query* de cinco con un tamaño de *tema_ usr* de once, mientras que la segunda combina un tamaño de *query* de uno con un tamaño de *tema_ usr* de tres.

La distinción entre las gráficas del lado derecho y del lado izquierdo radica en el tamaño del resultado de búsqueda, K. En las gráficas del lado izquierdo (combinación 6), el valor de K es uno, mientras que en las gráficas del lado derecho (combinación 8), el valor de K es cinco.

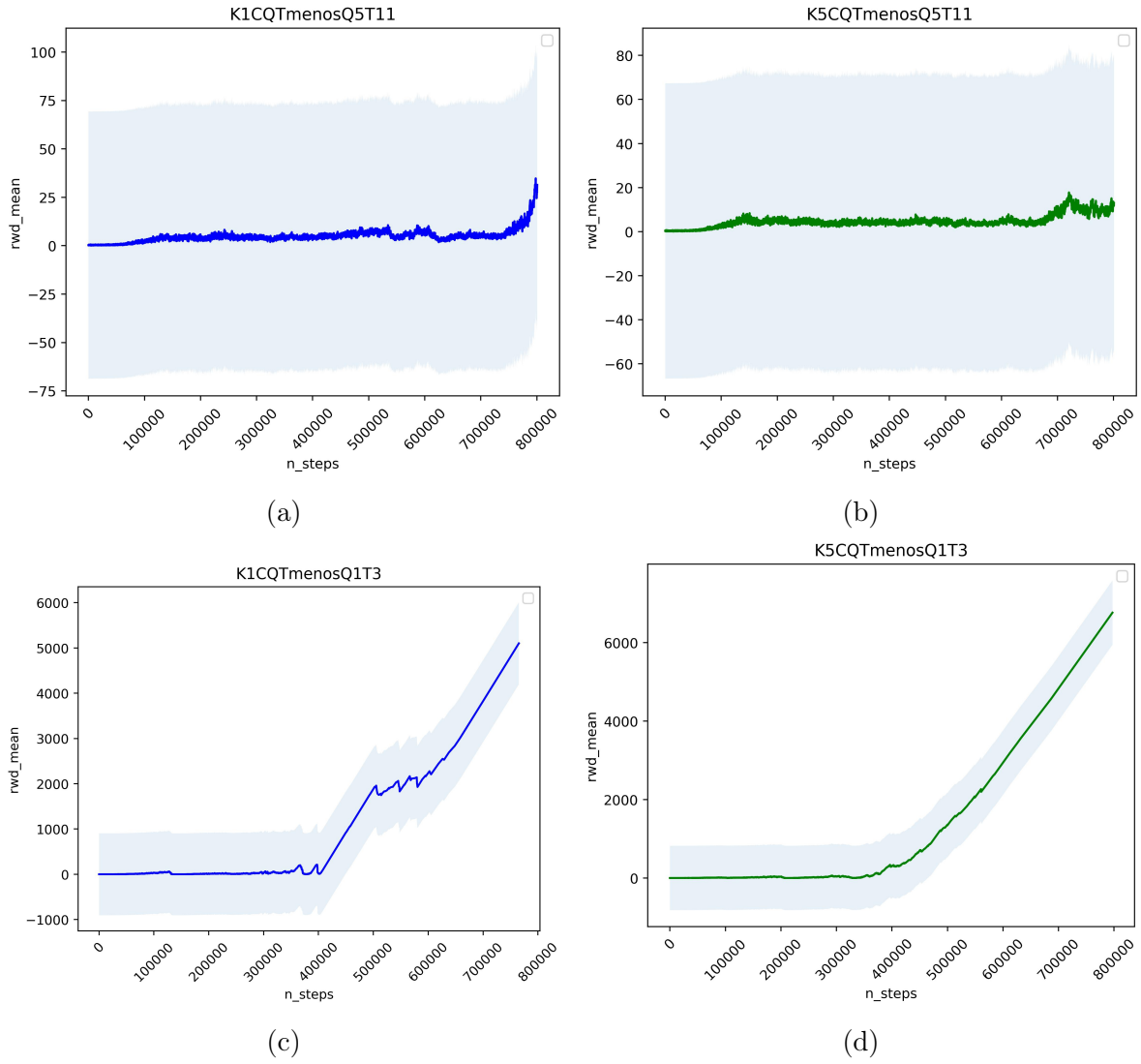


Figura 4.13: Tema menos preponderante y complejidad del cuartil inferior.

Las figuras 4.13a y 4.13b corresponden, respectivamente, a las combinaciones 6a y 8a de la tabla 4.3. En el transcurso del entrenamiento, la combinación 6a logró alcanzar un *rwd_mean* máximo de 35, mientras que la combinación 8a obtuvo un *rwd_mean* máximo de 17.5. A pesar de estos valores relativamente bajos de *rwd_mean*, ambas combinaciones exhibieron inestabilidad en su proceso de aprendizaje durante el entrenamiento.

Por otro lado, las figuras 4.13c y 4.13d corresponden, respectivamente, a las combinaciones 6b y 8b de la tabla 4.3. En estas gráficas, se puede

apreciar un aumento en el valor del *rwd_mean* a medida que se avanza en el entrenamiento, alcanzando finalmente valores máximos de 5,000 y 6,700 respectivamente al concluir el proceso de entrenamiento. Además de mostrar un mayor *rwd_mean*, las combinaciones 6b y 8b también presentaron una mayor estabilidad en la evolución de su curva de aprendizaje.

En la figura 4.14 se presentan cuatro gráficas correspondientes a las combinaciones 1 y 3 de la tabla 4.3. Las gráficas del lado derecho corresponden a la combinación 3, mientras que las gráficas del lado izquierdo corresponden a la combinación 1. Estas combinaciones comparten tres valores en común. El primer valor es el *tema_usr*, que es igual a Tmas. El segundo valor es la *complej_usr*, que es igual a CM. El tercer parámetro consta de dos combinaciones: la primera con un tamaño de *query* igual a cinco y un tamaño de *tema_usr* igual a once, y la segunda con un tamaño de *query* igual a uno y un tamaño de *tema_usr* igual a tres.

La diferencia principal entre las gráficas del lado derecho e izquierdo radica en el tamaño del resultado de búsqueda K. Para las gráficas del lado izquierdo o combinación 1, el valor de K es igual a uno. Por otro lado, en las gráficas del lado derecho o combinación 3, el valor de K es igual a cinco.

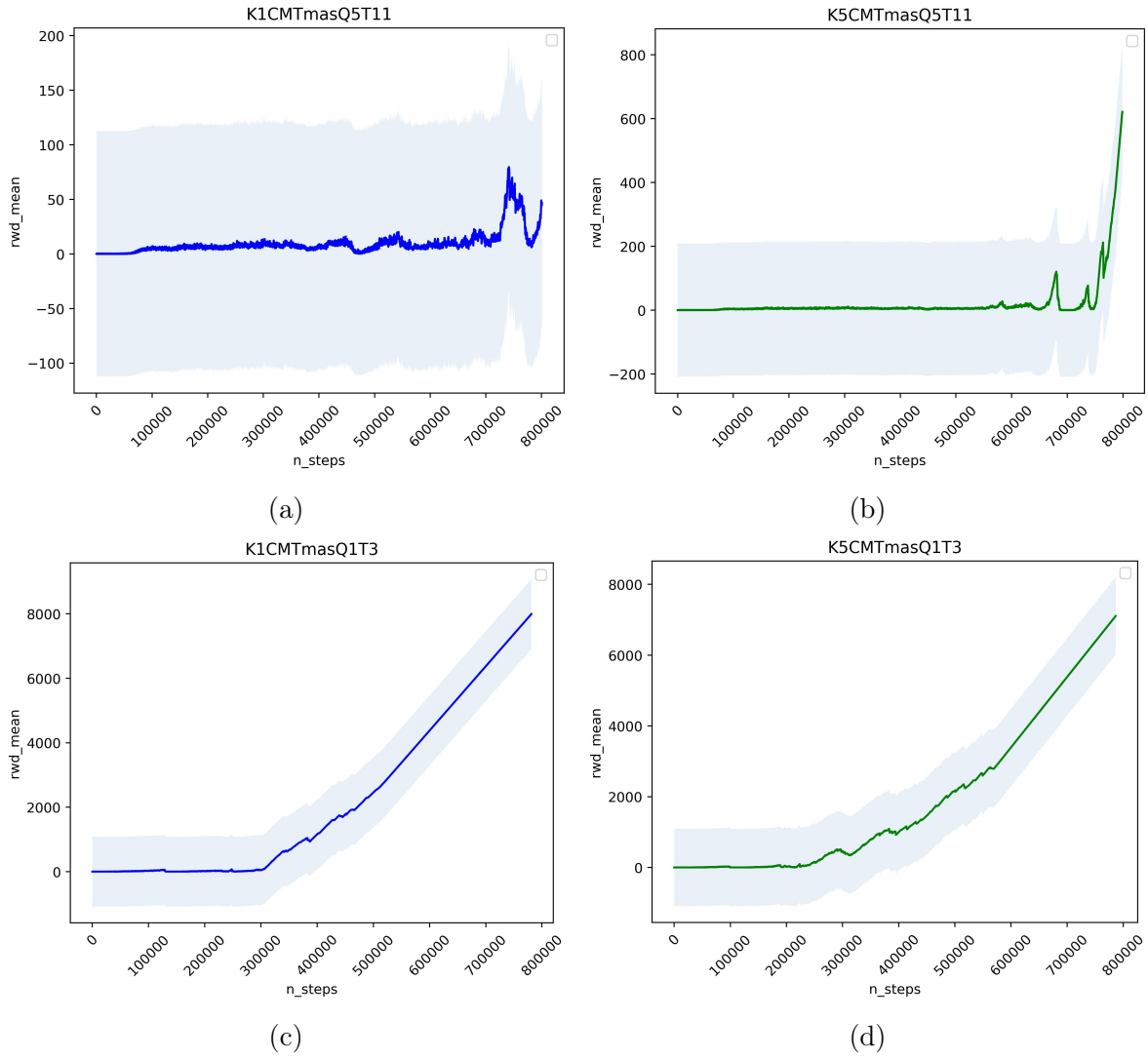


Figura 4.14: Tema más preponderante y complejidad mediana.

En las figuras 4.14a y 4.14b se pueden observar respectivamente las combinaciones 1a y 3a de la tabla 4.3. A lo largo del entrenamiento, ambas combinaciones alcanzaron un *rwd_mean* máximo aproximado de 80 y 600 respectivamente. Sin embargo, la combinación 1a exhibió poca *rwd_mean* y una notoria inestabilidad durante todo su proceso de entrenamiento. Por otro lado, la combinación 3a mostró una tendencia diferente. Al superar los 700K pasos, esta combinación presentó un incremento más significativo en su *rwd_mean*, acompañado de una mayor estabilidad en su crecimiento

a lo largo del tiempo.

Por otra parte, las figuras 4.14c y 4.14d representan respectivamente las combinaciones 1b y 3b de la tabla 4.3. En estas gráficas, se observa que a partir de los 300K pasos, ambos conjuntos empezaron a experimentar un aumento en el valor del *rwd_mean*, alcanzando máximos de 7,900 y 7,100 respectivamente al final del entrenamiento. Junto con su mayor *rwd_mean*, las combinaciones 1b y 3b demostraron una mayor estabilidad en el crecimiento de su curva de aprendizaje en comparación con las otras dos combinaciones anteriores.

La figura 4.15 presenta cuatro gráficas que corresponden a las combinaciones 2 y 4 de la tabla 4.3. En esta representación, las gráficas ubicadas en el lado derecho pertenecen a la combinación 4, mientras que las del lado izquierdo corresponden a la combinación 2. Estas combinaciones comparten tres valores en común. En primer lugar, el *tema_usr* es igual a Tmenos. En segundo lugar, la *complej_usr* es igual a CM. El tercer conjunto de parámetros consiste en dos combinaciones. La primera combina un tamaño de *query* de cinco junto con un *tema_usr* de once palabras. La segunda combina un tamaño de *query* de uno con un *tema_usr* de tres palabras.

La distinción principal entre las gráficas del lado derecho e izquierdo radica en el tamaño del resultado de búsqueda K. En las gráficas del lado izquierdo, es decir, la combinación 2, el valor de K es igual a uno. Mientras tanto, en las gráficas del lado derecho, que corresponden a la combinación 4, el valor de K es igual a cinco.

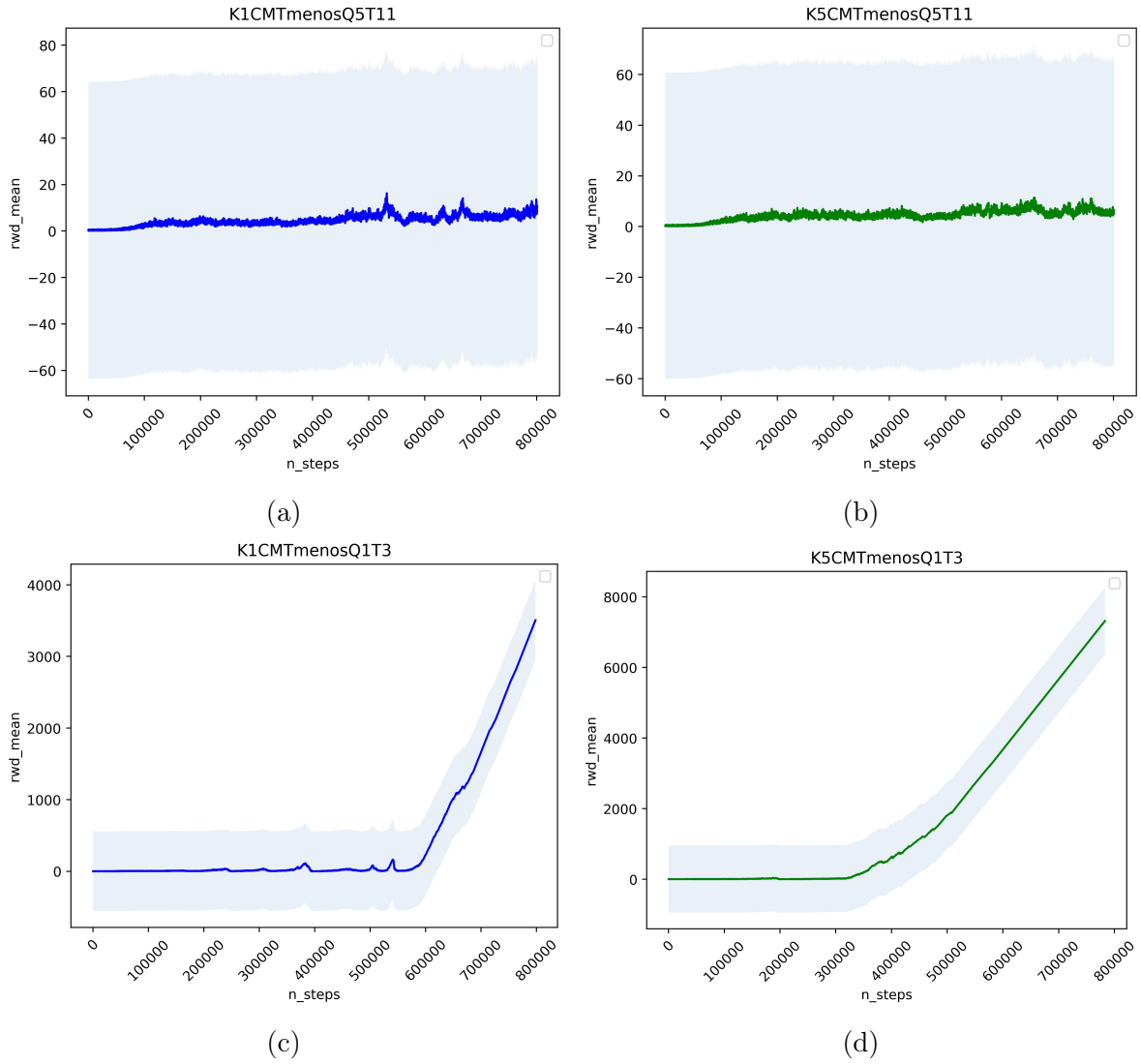


Figura 4.15: Tema menos preponderante y complejidad mediana.

Las figuras 4.15a y 4.15b representan, respectivamente, las combinaciones 2a y 4a de la tabla 4.3. A lo largo del proceso de entrenamiento, ambas combinaciones lograron alcanzar un *rwd_mean* máximo aproximado de 16 y 10, respectivamente. Sin embargo, además de evidenciar valores bajos de *rwd_mean*, estas combinaciones, es decir, 2a y 4a, mostraron una considerable inestabilidad durante el transcurso del entrenamiento.

Así mismo, las figuras 4.15a y 4.15b corresponden, respectivamente, a las combinaciones 2b y 4b de la tabla 4.3. En el caso de la combinación

2b, se observa un aumento gradual en el *rwd_mean* a medida que avanza el entrenamiento, llegando a alcanzar un valor máximo de 3,500 alrededor del paso 600K. En contraste, la combinación 4b exhibe un patrón similar, pero con un incremento más pronunciado en el *rwd_mean*, llegando a su punto máximo de 7,300 alrededor del paso 300K. Además de presentar un mayor *rwd_mean*, tanto las combinaciones 2b como 4b demostraron una mayor estabilidad en su curva de aprendizaje.

Errores gramaticales

Después de analizar el comportamiento del *rwd_mean* en los dieciséis experimentos a través de sus gráficas, es ahora pertinente examinar los errores gramaticales cometidos por el agente al realizar los *queries* durante estos experimentos. Por simplicidad, se les denominará como errores gramaticales o *e_gramar*. A continuación, se presentan en las siguientes gráficas el promedio de errores gramaticales de cada una de las combinaciones previamente mostradas.

La figura 4.16 exhibe las combinaciones 5a y 7a presentadas en la figura 4.12, según la información proporcionada en la tabla 4.3. En consecuencia, las combinaciones 5a y 7a se corresponden respectivamente con las figuras 4.16a y 4.16b. Como se mencionó previamente, estas combinaciones evidenciaron un bajo valor de *rwd_mean* durante el proceso de entrenamiento. Además, al considerar la cantidad de *e_gramar*, la combinación 5a presentó un total de 16,734, en contraste con la combinación 7a que registró un total de 19,547. De este modo, es notorio que tanto las combinaciones 5a como 7a, además de poseer un *rwd_mean* inferior, también exhiben un alto número de errores gramaticales, tal como se aprecia en las representaciones gráficas de sus promedios.

En contraste, las figuras 4.16c y 4.16d corresponden a las combinaciones 5b y 7b de acuerdo con la tabla 4.3. Estas combinaciones exhibieron un *rwd_mean* superior en comparación con las combinaciones 5a y 7a. Con respecto al número de *e_gramar*, la combinación 5b acumuló un total de 346, mientras que la combinación 7b registró 382. En consecuencia, las combinaciones 5b y 7b no solo presentaron un *rwd_mean* más alto, sino que también demostraron mayor estabilidad en la reducción de los *e_gramar*, tal como se puede apreciar en las gráficas de sus medias.

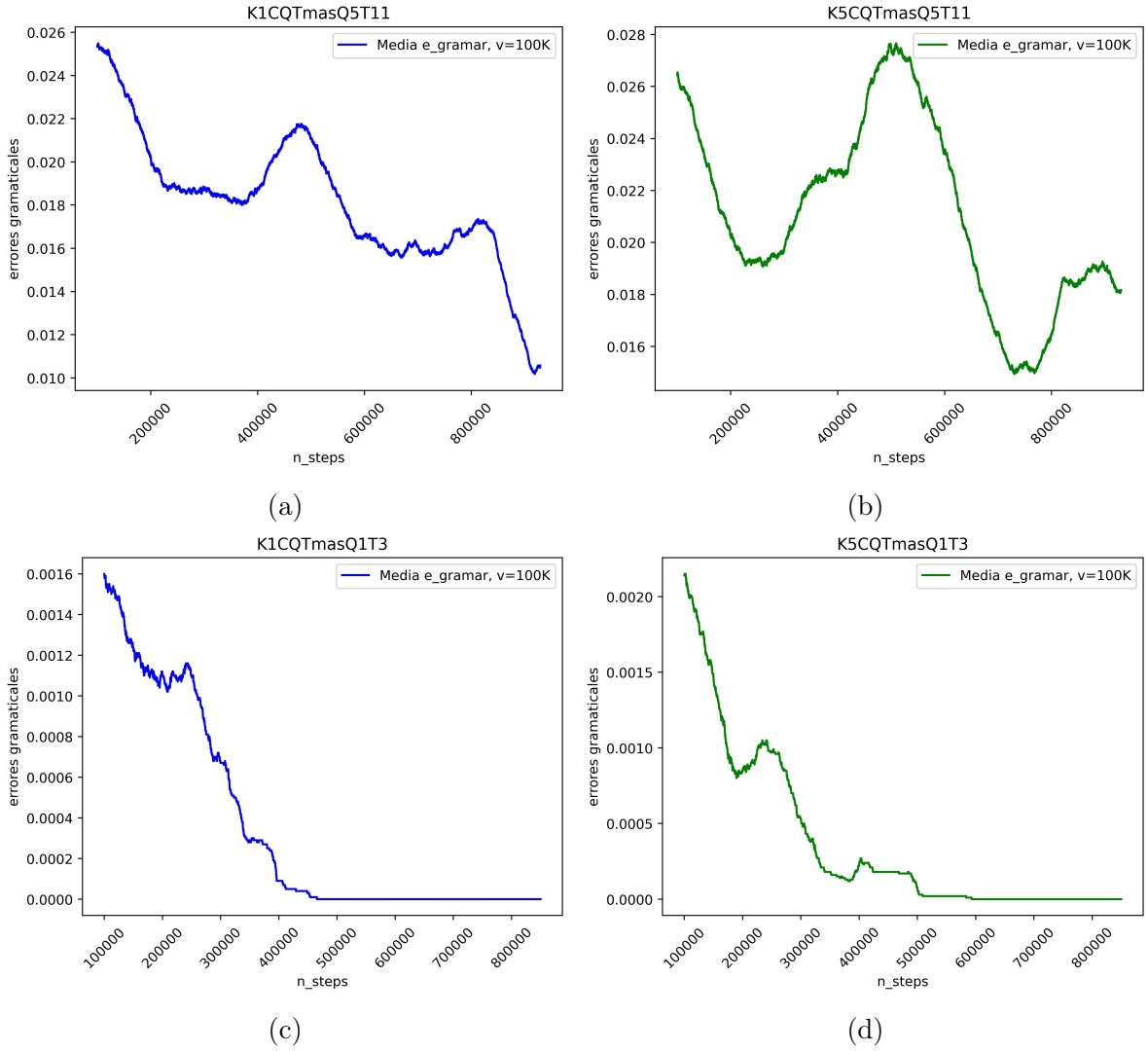


Figura 4.16: Tema más preponderante y complejidad del cuartil inferior.

Siguiendo con las combinaciones subsiguientes, en la figura 4.17 se puede observar que las combinaciones 6a y 8a se corresponden con las figuras 4.17a y 4.17b, respectivamente. Como se mencionó previamente, estas configuraciones exhibieron un *rwd_mean* bajo durante el entrenamiento. Ahora, en términos del número de errores gramaticales *e_gramar*, la combinación 6a acumuló un total de 19,227, mientras que la combinación 8a registró 20,182. Por lo tanto, además de tener un *rwd_mean* inferior, las combinaciones 6a y 8a evidenciaron una mayor inestabilidad en la reducción de los errores gramaticales, como se refleja en las gráficas de sus promedios.

En contraste, las figuras 4.17c y 4.17d se asocian respectivamente a las combinaciones 6b y 8b de la tabla 4.3. Estas configuraciones presentaron un *rwd_mean* más elevado en comparación con las combinaciones 6a y 8a. En lo que respecta al número de *e_gramar*, la combinación 6b registró un total de 289, mientras que la combinación 8b reportó 292. Por lo tanto, las combinaciones 6b y 8b no solo demostraron un *rwd_mean* superior, sino que también exhibieron una mayor estabilidad en la reducción de los *e_gramar*, tal como se evidencia en las gráficas de sus medias.

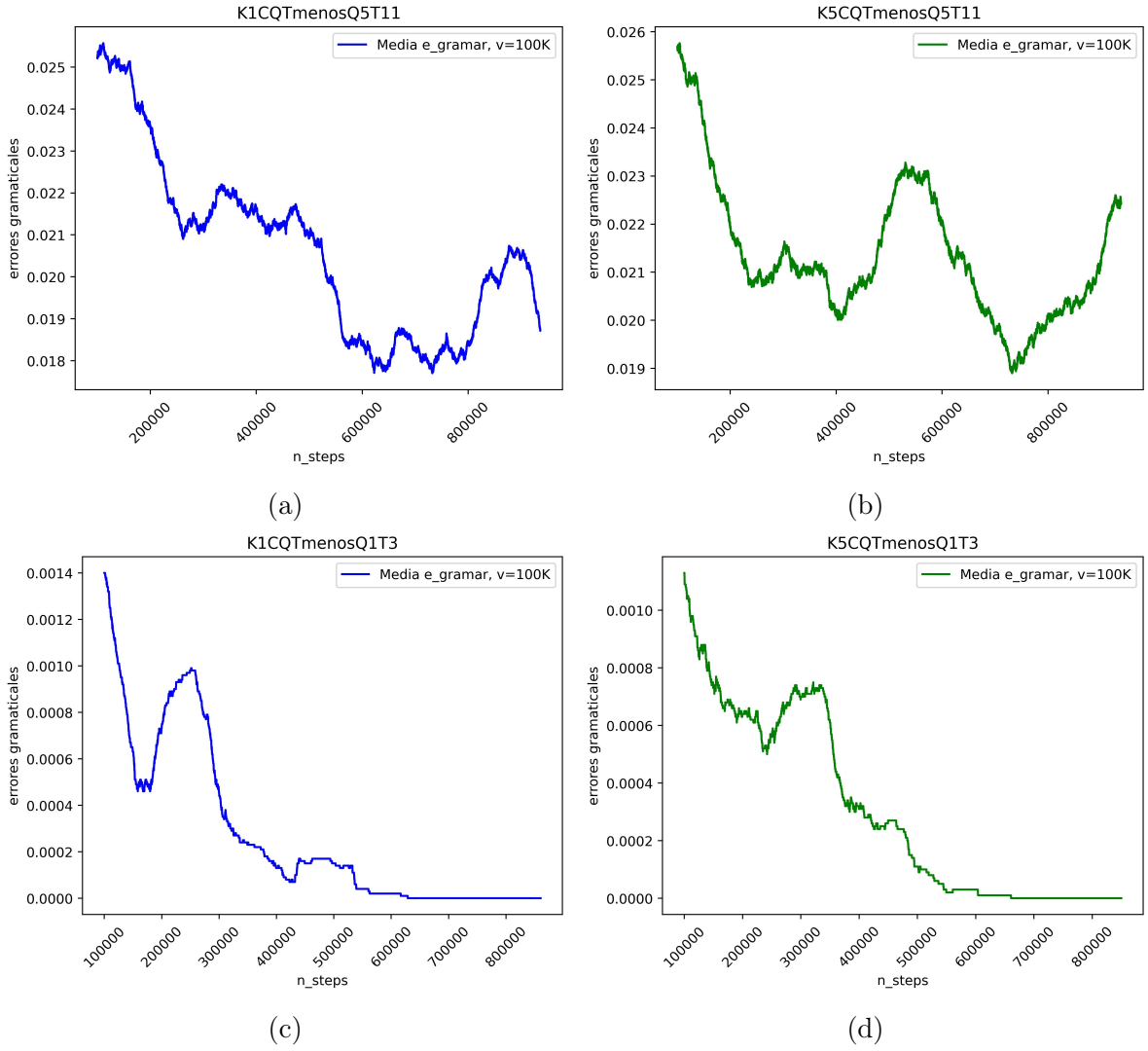


Figura 4.17: Tema menos preponderante y complejidad del cuartil inferior.

Siguiendo con los próximos cuatro experimentos, se puede apreciar en la figura 4.18 que las combinaciones 1a y 3a se correlacionan respectivamente con las figuras 4.18a y 4.18b. Como se ha mencionado previamente, estas combinaciones presentaron un *rwd_mean* bajo durante el proceso de entrenamiento. Ahora, centrándonos en el recuento de *e_gramar*, la combinación 1a registró un total de 19,323, mientras que la combinación 3a reportó 19,180. Por lo tanto, además de un *rwd_mean* inferior, las combinaciones 1a y 3a muestran una mayor inestabilidad en la reducción de los

e_gramar, tal como se observa en las gráficas de sus medias.

Por otra parte, las figuras 4.18c y 4.18d corresponden respectivamente a las combinaciones 1b y 3b de la tabla 4.3. Estas combinaciones lograron un *rwd_mean* más alto en comparación con las combinaciones 1a y 3a. En términos del recuento de *e_gramar*, la combinación 1b registró un total de 260, lo que la convierte en la combinación con el menor número de *e_gramar* de todas. En contraste, la combinación 3b tuvo un recuento de 343. Por lo tanto, las combinaciones 1b y 3b, además de presentar un *rwd_mean* superior, también exhibieron una mayor estabilidad en la reducción de los *e_gramar*, tal como se aprecia en las gráficas de sus medias.

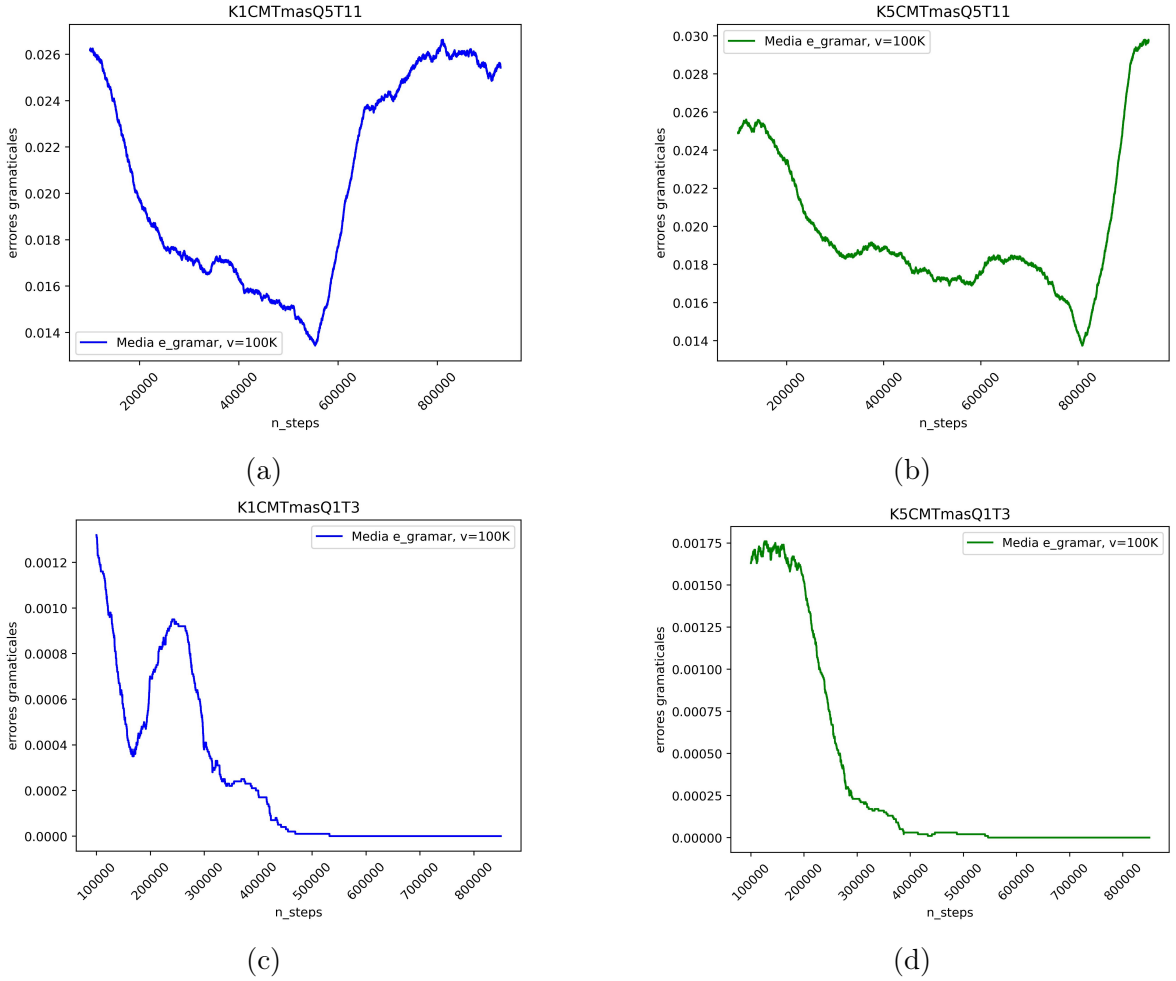


Figura 4.18: Tema más preponderante y complejidad mediana.

Los resultados de los últimos cuatro experimentos se presentan en la figura 4.19. En esta representación gráfica, se encuentran las combinaciones 2a y 4a, que corresponden a las figuras 4.19a y 4.19b, respectivamente. Tal como se mencionó previamente, estas combinaciones presentaron un *rwd_mean* reducido durante el entrenamiento. Analizando ahora el recuento de errores gramaticales *e_gramar*, la combinación 2a tuvo un total de 20689, posicionándose como la combinación con el mayor número de *e_gramar* entre todas las variantes. En contraste, la combinación 4a registró un recuento de 19808. Por lo tanto, además de su menor *rwd_mean*, las combinaciones 2a y 4a evidencian mayor inestabilidad en la disminución de los *e_gramar*, como se puede observar en las gráficas de sus medias.

En contrapartida, las figuras 4.19c y 4.19d representan respectivamente las combinaciones 2b y 4b de la tabla 4.3. Estas variantes obtuvieron valores más elevados de *rwd_mean* en comparación con las combinaciones 2a y 4a. En lo que respecta al conteo de *e_gramar*, la combinación 2b acumuló un total de 486, mientras que la combinación 4b presentó un conteo de 438. Como resultado, las combinaciones 2b y 4b, además de exhibir un *rwd_mean* superior, también mostraron una mayor estabilidad en la reducción de los *e_gramar*, como se observa en las gráficas que representan sus medias.

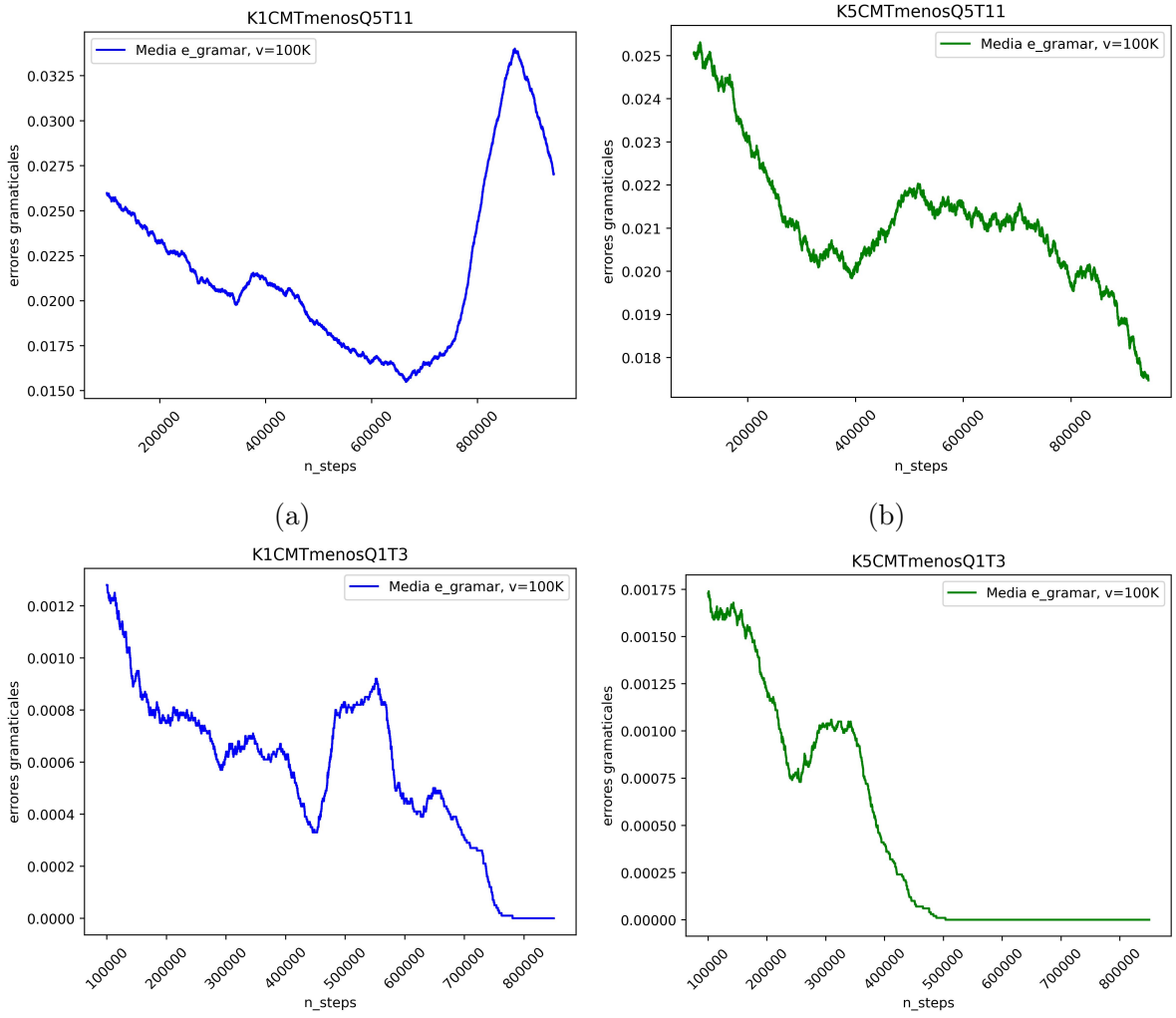


Figura 4.19: Tema menos preponderante y complejidad mediana.

Los experimentos se han ejecutado utilizando las dieciséis combinaciones detalladas en la tabla 4.3. A través de estas configuraciones y las adaptaciones efectuadas en el Experimento 3, se han identificado los ocho experimentos con los valores de *rwd_mean* más elevados, presentados en orden descendente en la tabla 4.4.

Tabla 4.4: Resultados con mayor *rwd_mean*

K	**Complejidad**	**Tema**	**Query**	**Longitud Tema**	**rwd_mean**	**e_gramar**
K1	CQ	Tmas	Q1	T3	8000	346
K1	CM	Tmas	Q1	T3	7900	260
K5	CM	Tmenos	Q1	T3	7300	438
K5	CM	Tmas	Q1	T3	7100	343
K5	CQ	Tmenos	Q1	T3	6700	292
K5	CQ	Tmas	Q1	T3	6100	382
K1	CQ	Tmenos	Q1	T3	5000	289
K1	CM	Tmenos	Q1	T3	3500	486

El parámetro *e_gramar* que se presenta en la tabla 4.4 refleja la cantidad de errores gramaticales detectados en todos los *queries* generados por el agente en cada uno de los experimentos. Basándonos en los resultados obtenidos en estos ocho experimentos, se pueden extraer las siguientes conclusiones:

1. De los 16 experimentos realizados en el Experimento 3, los que lograron obtener el mayor *rwd_mean* por arriba de 1000 fueron las ocho combinaciones mostradas en la tabla 4.4.

2. El primer factor que contribuyó a obtener el mayor *rwd_mean* fue la combinación de un *query* de longitud igual a uno y un *tema_usr* de longitud igual a tres. La longitud tanto del *query* como del *tema_usr* tiene un impacto significativo en la carga computacional requerida para llevar a cabo las operaciones. Un *query* de tamaño cinco y un *tema_usr* de tamaño once, por ejemplo, implica un mayor nivel de complejidad y dificultad para encontrar similitudes con las conversaciones presentes en la base de datos.

3. El segundo factor que contribuyó a obtener el mayor *rwd_mean* es la preponderancia del *tema_usr*. En la base de datos, algunas palabras

aparecen con mayor frecuencia que otras. Como se puede apreciar en la tabla 4.4, el *tema_usr* igual a *Tmas* mostró un *rwd_mean* más alto en comparación con *Tmenos*. Esto sugiere que la elección de un *tema_usr* que contiene palabras más frecuentes en las conversaciones puede tener un impacto positivo en el rendimiento del agente.

4. El tercer factor que contribuyó a obtener un mayor *rwd_mean* en estos ocho experimentos es la cantidad de resultados K que el buscador devuelve al entorno. Como se muestra en la tabla 4.4, la configuración con K=1 obtuvo el mayor *rwd_mean* solo para *Tmas*. Por otro lado, con K=5, la configuración que tuvo el mejor *rwd_mean* fue *Tmenos*. Esto está relacionado con el parámetro *simi_mediana*, ya que la mediana representa el valor central en un conjunto de datos ordenados de forma ascendente o descendente.

   Cuando se tiene la combinación de *Tmas* y K=1, la *simi_mediana* tiende a ser mayor, ya que ese valor es el más preponderante en la muestra de resultados. En cambio, cuando se tiene la combinación de *Tmas* y K=5, la *simi_mediana* tiende a ser menor, ya que los valores subsiguientes son menos preponderantes.

   Por otro lado, cuando se utiliza la combinación de *Tmenos* y K=1, la *simi_mediana* tiende a ser menor, ya que ese valor es el menos preponderante en la muestra de resultados. En cambio, cuando se tiene la combinación de *Tmenos* y K=5, la *simi_mediana* tiende a ser mayor, ya que los valores anteriores son más preponderantes en la muestra de resultados.

   En resumen, la elección de la cantidad de resultados K puede influir en la *simi_mediana* y, por lo tanto, en el rendimiento del agente, dependiendo de la preponderancia del *tema_usr* y la distribución de similitudes en la base de datos.

5. El cuarto factor que contribuyó a obtener el mayor *rwd_mean* son las *complej_usr*. De acuerdo con la tabla 4.4, la combinación con K=1, *CQ* y *Tmas* obtiene un *rwd_mean* de 8000, que es el más alto de todos los experimentos. Sin embargo, cuando a esta combinación se le modifica el valor de K=5, el *rwd_mean* disminuye a 6100, ubicándose como la sexta mejor combinación en términos de *rwd_mean*. Esta disminución se debe al mismo efecto de la mediana, como se describió anteriormente con las similitudes, pero ahora aplicado a las complejidades.

   La siguiente combinación es K=1, *CM* y *Tmas*, la cual obtuvo un *rwd_mean* de 7900, siendo la segunda mejor en todos los experimentos. Por otro lado, cuando a esta combinación se le modifica K=5 y se calculan las *complej_medianas*, el *rwd_mean* disminuye a 7100, ubicándose como la cuarta mejor combinación en términos de *rwd_mean*. De las cuatro combinaciones con *Tmas*, la que acumuló el mayor *rwd_mean* fue aquella que incluyó las complejidades *CM*. Esto sugiere que *Tmas* tiene ligeramente mayor presencia en *CM* en comparación con *CQ*.

   Ahora, para los temas menos preponderantes, la combinación K=5, *CM* y *Tmenos* obtuvo un *rwd_mean* de 7300, siendo la tercera mejor en términos de *rwd_mean* en todos los experimentos. Por otra parte, cuando a esta combinación se le modifica K=1, el *rwd_mean* disminuye drásticamente a 3500, ubicándose como la más baja de estas ocho combinaciones.

   Por otro lado, la quinta combinación que alcanzó un *rwd_mean* de 6700 incluye K=5, *CQ* y *Tmenos*. Sin embargo, al modificar el valor de K=1, el *rwd_mean* disminuye a 5000, ubicándose como la séptima mejor combinación en términos de *rwd_mean*. De las cuatro combinaciones con *Tmenos*, la que acumuló el mayor *rwd_mean* fue aquella

que incluyó las complejidades *CQ*. Esto sugiere que *Tmenos* tiene una mayor presencia de complejidades en *CQ* en comparación con *CM*.

6. De acuerdo con las observaciones realizadas en el punto cinco, se puede concluir que de los ocho experimentos que obtuvieron el mayor *rwd_mean*, únicamente aquellos que presentaron temas más preponderantes (Tmas) y complejidades medianas (*CM*) lograron el mayor *rwd_mean*. Esto sugiere que tanto *Tmas* como *CM* contienen información más relevante o frecuente en la base de datos. En otras palabras, el agente obtuvo mejores resultados cuando se enfocó en temas más comunes y complejidades medianas en lugar de temas menos preponderantes o complejidades más básicas (*CQ*). Estos hallazgos resaltan la importancia de considerar la prevalencia de los temas y las complejidades al ajustar los parámetros para mejorar el rendimiento del agente en la recuperación de respuestas relevantes.

En relación a la tabla 4.4, que exhibió las ocho combinaciones con el mayor *rwd_mean*, estas mismas ocho combinaciones se presentan ahora en la tabla 4.5, pero esta vez se ordenan de menor a mayor en términos de errores gramaticales.

Tabla 4.5: Resultados con menor *errores gramaticales*

K	**Complejidad**	**Tema**	**Query**	**Longitud Tema**	**rwd_mean**	**e_gramar**
K1	CM	Tmas	Q1	T3	7900	260
K1	CQ	Tmenos	Q1	T3	5000	289
K5	CQ	Tmenos	Q1	T3	6700	292
K5	CM	Tmas	Q1	T3	7100	343
K1	CQ	Tmas	Q1	T3	8000	346
K5	CQ	Tmas	Q1	T3	6100	382
K5	CM	Tmenos	Q1	T3	7300	438
K1	CM	Tmenos	Q1	T3	3500	486

De acuerdo con la tabla 4.5, mencionamos las siguientes conclusiones:

1. La combinación *K1*, *CM* y *Tmas*, que obtuvo el segundo mejor *rwd_mean*, registró el menor número de *e_gramar* en comparación con todos los experimentos. Esto se debe a la alta frecuencia de los parámetros *Tmas* y *CM* en la base de datos.

2. Como se ha mencionado previamente, la combinación *K1*, *CQ* y *Tmas* fue la que obtuvo mayor *rwd_mean*, aunque también presentó un mayor número de *e_gramar*. Esto se debe a que los *Tmas* tienen una relación menos estrecha con *CQ* en la base de datos.

3. Por otro lado, la combinación *K1*, *CM* y *Tmenos* fue la que logró el menor *rw_mean* de estos ocho experimentos, y al mismo tiempo, presentó el mayor número de *e_gramar*. Esto se debe a que los *Tmenos* son pocos frecuentes en la base de datos y además, no guardan una relación tan estrecha con las *CM*.

Continuando con los ocho experimentos restantes, en la tabla 4.6 se presentan las combinaciones que obtuvieron el menor *rwd_mean*.

Tabla 4.6: Resultados con menor *rwd_mean*

K	**Complejidad**	**Tema**	**Query**	**Longitud Tema**	**rwd_mean**	**e_gramar**
K5	CM	Tmas	Q5	T11	600	19180
K1	CM	Tmas	Q5	T11	80	19323
K1	CQ	Tmas	Q5	T11	35	16734
K1	CQ	Tmenos	Q5	T11	35	19227
K5	CQ	Tmas	Q5	T11	30	19547
K5	CQ	Tmenos	Q5	T11	17.5	20182
K1	CM	Tmenos	Q5	T11	16	20689
K5	CM	Tmenos	Q5	T11	10	19808

En la tabla 4.6 se observa que el *rwd_mean* están ordenadas de mayor a menor. Las conclusiones al realizar estos ocho experimentos restantes son las siguientes:

1. Las últimas ocho combinaciones con menor *rwd_mean* comparten el tamaño del *query* igual a cinco y la longitud del *tema_usr* igual a once. Esto, como se mencionó anteriormente, se debe a la capacidad de cómputo necesaria para realizar más operaciones con un número mayor de parámetros, lo que afecta negativamente el rendimiento de la búsqueda y la generación de respuestas adecuadas.

2. Los dos experimentos con el mayor *rwd_mean* tienen en común la prevalencia de los parámetros *Tmas* y la complejidad de usuario *CM*. Esto se debe a que estos dos parámetros están estrechamente relacionados y son más frecuentes en la base de datos, lo que facilita la generación de respuestas efectivas y coherentes por parte del sistema.

3. Las dos combinaciones que tuvieron el menor *rwd_mean* de los dieciséis experimentos son las que comparten el *Tmenos* y la complejidad *CM*. Se observa que esta combinación es poco frecuente en la base de datos.

Con referencia a la tabla 4.6, donde se presentaron las ocho combinaciones con el menor *rwd_mean*, estas mismas ocho combinaciones se muestran ahora en la tabla 4.7, pero esta vez ordenadas de menor a mayor en cuanto a los *e_gramar*.

Tabla 4.7: Resultados con mayor *errores gramaticales*

K	**Complejidad**	**Tema**	**Query**	**Longitud Tema**	**rwd_mean**	**e_gramar**
K1	CQ	Tmas	Q5	T11	35	16734
K5	CM	Tmas	Q5	T11	600	19180
K1	CQ	Tmenos	Q5	T11	35	19227
K1	CM	Tmas	Q5	T11	80	19323
K5	CQ	Tmas	Q5	T11	30	19547
K5	CM	Tmenos	Q5	T11	10	19808
K5	CQ	Tmenos	Q5	T11	17.5	20182
K1	CM	Tmenos	Q5	T11	16	20689

1. Se puede observar una disminución en la cantidad de *e_gramar* en los *Tmas*. Esta reducción tiene sentido cuando comparamos una consulta de cinco palabras con la base de datos, ya que es probable que haya menos *e_gramar* al intentar identificar las palabras más frecuentes en dicha base de datos.

2. Por otra parte, la perspectiva opuesta consiste en buscar las palabras menos frecuentes en la base de datos. Esta tarea se torna más complicada cuando la consulta consta de cinco palabras. En consecuencia, el agente debe invertir un esfuerzo considerable al intentar localizar palabras poco comunes en la base de datos.

4.1.3. Discusión

4.1.3.1. Experimento 1

En la Figura 4.8, se puede observar un pico o sobresalto en la gráfica que llamó la atención. Este fenómeno se debe al hecho de que el agente comenzó a emitir consultas con un nivel de cercanía al tema y complejidad del usuario muy elevado, lo que dio lugar a una acumulación más significativa de recompensas durante este período de tiempo en la escala.

4.1.3.2. Experimento 2

Durante el Experimento 2, se observaron gráficas con valores más altos de *rwd_mean*. Sin embargo, la mayoría de estas gráficas exhibieron inestabilidad. Esto plantea la pregunta: ¿Puede la combinación del *query*, el *tema_usr* y la *complej_usr* influir en la estabilidad de *rwd_mean*?

Además, es importante considerar si las limitaciones de la base de datos

tuvieron algún impacto en el comportamiento del agente.

4.1.3.3. Experimento 3

Al finalizar el Experimento 3, se observó un aumento en el valor de *rwd_mean*, que pasó de un máximo de 6,000 en el Experimento 2 a un máximo de 8,000 en el Experimento 3. Sin embargo, es importante notar que en el Experimento 2, este valor máximo de *rwd_mean* se alcanzó utilizando el *query* máximo, que tenía un tamaño de cuatro. En contraste, en el Experimento 3, el valor máximo de *rwd_mean* se obtuvo con el *query* más pequeño, que tenía un tamaño de uno. Este cambio en el *rwd_mean* está relacionado tanto con el tamaño de la base de datos como con el tamaño del *query*. En otras palabras, buscar cinco palabras en una base de datos más pequeña resulta ser más eficiente que hacerlo en una de mayor tamaño.

4.2. Predicciones con el mejor modelo evaluado

4.2.1. Requisitos para llevar a cabo las predicciones

Después de llevar a cabo las evaluaciones en cada uno de los entrenamientos durante el Experimento 3, el siguiente paso consiste en utilizar el modelo que haya demostrado el mejor desempeño, medido en términos de un mayor *rwd_mean* y un menor número de *e_gramar*. Este modelo seleccionado se utilizará para realizar predicciones.

Para poder realizar las predicciones correspondientes, es necesario contar con los siguientes elementos:

Modelo mejor evaluado

El modelo PPO que demostró el mejor desempeño y que se utilizará para llevar a cabo las predicciones es aquel que presenta la combinación K1CMTmasQ1T3.

Entorno

Para llevar a cabo las predicciones, se utilizarán como parámetros una instancia de la clase Entorno y el método vectorizador que se aplicaron durante el Experimento 3. El tema de usuario que se ingresará es *Tmas*, con un tamaño de tres palabras, correspondiente a 'school english students', y una complejidad *CM* igual a 0.115.

Base de datos

La nueva base de datos que se utilizará para realizar las predicciones está compuesta por un total de 400 conversaciones de acceso libre disponibles en la web [1].

Con la integración de todos estos elementos y mediante la implementación de un pequeño algoritmo en Python, contamos con todo lo necesario para llevar a cabo las predicciones y obtener los resultados deseados.

4.2.2. Resultados de las predicciones

Utilizando un bucle *for* de 1,000 pasos, se generaron 1,000 predicciones, de las cuales únicamente seis lograron un *rwd* igual a 1. Estos resultados se representan en la Figura 4.20, que muestra la media móvil con un intervalo de confianza del 95 % del *rwd* obtenido durante las predicciones.

[1] https://www.esl-lab.com/

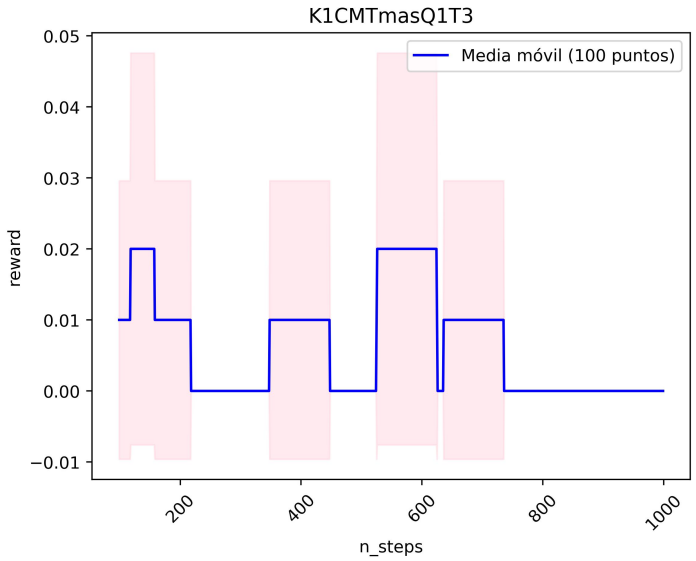


Figura 4.20: Gráfica de rwd_mean con predicción de 1,000 pasos

De las 1,000 predicciones efectuadas en la nueva base de datos, es relevante destacar que en los *queries* realizados por el modelo no se registraron *e_gramar*.

4.2.3. Discusión

Al concluir las predicciones con el mejor modelo evaluado previamente en los tres experimentos, se obtuvieron solo seis conversaciones cercanas a los requerimientos del usuario. Las preguntas con base en estos resultados, son las siguientes:

- ¿Los pocos resultados que se obtuvieron dependió de la configuración de los hiperparámetros del agente como: tasa de aprendizaje, el tamaño del lote (batch size), el número de capas, número de nodos, la política, etc.?
- Utilizar otro modelo de agente, por ejemplo el modelo A2C, ¿hubiera mejorado el rendimiento del sistema?

- De igual forma, ¿La configuración de los parámetros del entorno afectaron los pocos resultados arrojados por el modelo?. Por ejemplo, la configuración de los umbrales.
- ¿La situación previamente mencionada condujo a que el modelo se enfocara en la explotación en lugar de continuar con la exploración?
- ¿Hubiera mejorado el rendimiento si se hubiera utilizado una base de datos con un mayor número de conversaciones?

Durante el desarrollo y la fase final del proyecto, surgieron estos cuestionamientos y dudas en relación a cada uno de los resultados obtenidos.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

En este trabajo de tesis, se implementó una herramienta de búsqueda personalizada con el propósito de mejorar la práctica auditiva del inglés, utilizando un agente neuronal. Este sistema ha cumplido con éxito los objetivos establecidos durante su fase de diseño, destacándose especialmente por alcanzar un rendimiento superior en cada ajuste realizado. Además, durante la etapa de predicciones, el sistema proporcionó opciones para seleccionar más de una conversación, todas ellas ajustadas a los requisitos de similitud y complejidad de vocabulario deseados por el estudiante.

Este buscador personalizado ha contribuido significativamente al estado del arte en su tarea específica de búsqueda. Sus parámetros de búsqueda, como la temática y la complejidad de vocabulario, rara vez se consideraban al buscar material adecuado para la práctica de la habilidad auditiva en inglés. Además, el agente neuronal, a través de sus acciones en el sistema de recuperación de información, se orienta hacia el contenido más relevante para el tema del usuario. Esto proporciona una ventaja significativa para

los usuarios, ya que el agente puede realizar búsquedas incluso cuando el usuario no conoce las palabras precisas para formular la consulta.

Los experimentos realizados en esta tesis confirmaron la hipótesis que sustentamos, la cual afirma que es posible construir un motor de búsqueda personalizado para la práctica del inglés mediante la implementación de un agente neuronal como control de un método de recuperación de información. Una parte fundamental de este sistema es el bloque entorno, el cual se diseñó tomando como base el funcionamiento del sistema CartPole, que es un sistema ampliamente probado en el campo del aprendizaje por refuerzo. Fue en este bloque donde se definieron los parámetros característicos de los sistemas de aprendizaje por refuerzo, como los estados y el cálculo de la recompensa. Además, se aplicó el principio de Markov en la definición de los estados o vector de observaciones.

Finalmente, la conclusión de este buscador personalizado sienta las bases para el desarrollo de nuevas herramientas en el área de las lenguas, en donde no solo se tome en consideración la estructura de aprendizaje de la lengua, ni el diseño tecnológico de la herramienta, sino también considere la parte del conocimiento del usuario y las cosas que ignore como el uso correcto de palabras claves para realizar una búsqueda efectiva.

5.2. Trabajo futuro

5.2.1. Experimentos

Con base en los resultados expuestos en este proyecto de tesis, se puede vislumbrar que el motor de búsqueda personalizado tiene un potencial considerable para mejorar en términos del número de resultados que ofre-

ce. Para lograr esta optimización, se proponen experimentos con diversos hiperparámetros tanto del agente como del sistema de recuperación de información. El objetivo es obtener recompensas más favorables, lo que se traduciría en una exploración más extensa de la base de datos y, por ende, en una mejora de las respuestas brindadas al usuario.

Además, se pueden realizar otros ajustes como la variación del tema de usuario. Esto implica no solo definir los temas más o menos predominantes en la base de datos, sino también establecer de manera aleatoria diferentes tópicos para evaluar el rendimiento del agente. Del mismo modo, la complejidad del usuario puede ser aleatoria, lo que permitiría observar los resultados que el agente produce en diferentes contextos.

5.2.2. Complejidad cognitiva

En este proyecto, se consideró únicamente la complejidad de vocabulario en las conversaciones. No obstante, en el campo del análisis de texto y la lingüística computacional, existen otros tipos de complejidades, como la complejidad cognitiva. Este enfoque se encarga de analizar los textos en busca de diversos parámetros cognitivos en párrafos y oraciones de los documentos. Algunos ejemplos de estos parámetros cognitivos incluyen palabras relacionadas con el pensamiento, certeza, duda, causa-efecto, entre otros.

La complejidad cognitiva podría ser considerada como otro tipo de complejidad a evaluar en las conversaciones. Esto permitiría identificar cuáles conversaciones contienen una mayor dificultad cognitiva y cuáles son más fáciles de comprender para el usuario. Aplicar esta perspectiva adicional de complejidad a la base de datos podría arrojar resultados interesantes en los experimentos.

5.2.3. Interfaz

En cuanto a la interacción con el usuario de una manera más amigable, se plantea como trabajo futuro el desarrollo de una interfaz de usuario. El propósito de esta interfaz es facilitar la accesibilidad de la herramienta a diferentes tipos de usuarios que deseen utilizar el buscador personalizado. Esto permitirá que la herramienta sea más fácil, útil, agradable e intuitiva de utilizar, en lugar de requerir conocimientos en código y comandos.

Bibliografía

[1] M. d. R. Carranza Alcántar, C. Islas Torres y M. L. Maciel Gómez, "Percepción de los estudiantes respecto del uso de las TIC y el aprendizaje del idioma inglés", *Apertura (Guadalajara, Jal.)*, vol. 10, n.º 2, págs. 50-63, 2018.

[2] S. Rahbar y S. Khodabakhsh, "English songs as an effective asset to improve listening comprehension ability; Evidence from Iranian EFL learners", *International Journal of Applied Linguistics and English Literature*, vol. 2, n.º 6, págs. 63-66, 2013.

[3] B. Liu, X. Lu y J. S. Culpepper, "Strong natural language query generation", *Information Retrieval Journal*, vol. 24, n.º 4, págs. 322-346, 2021.

[4] M.-H. Hsu, "A personalized English learning recommender system for ESL students", *Expert Systems with Applications*, vol. 34, n.º 1, págs. 683-688, 2008.

[5] M. Gao, J. Xing, C. Yin y L. Dai, "Personalized recommendation method for English teaching resources based on artificial intelligence technology", en *Journal of Physics: Conference Series*, IOP Publishing, vol. 1757, 2021, pág. 012 104.

[6] Z. Zulfikar, C. T. Aulia y S. Akmal, "Exploring Efl Students' Problems in Listening To English News Broadcasts", *Language Literacy: Journal of Linguistics, Literature, and Language Teaching*, vol. 4, n.º 2, págs. 340-352, 2020.

[7] C. of Europe. Council for Cultural Co-operation. Education Committee. Modern Languages Division, *Common European framework of reference for languages: Learning, teaching, assessment*. Cambridge University Press, 2001.

[8] M. Shokri, H. R. Tizhoosh y M. Kamel, "Reinforcement learning for personalizing image search", *LORNET Annu. E-Learning*, 2006.

[9] B. Ríos Miranda y W. E. Portugal Durán, "Análisis de tutores inteligentes como sustento en la Universidad Mayor de San Andrés", *Educación Superior*, vol. 6, n.º 2, págs. 37-46, 2019.

[10] www.cambridgeenglish.org., *Cambridge English, "International Language Standards: About the Common European Framework of Reference for Languages"*, Last accessed 28 June 2022, 2022. dirección: `https://www.cambridgeenglish.org/exams-and-tests/cefr/`.

[11] A. Capel y W. Sharp, *Objective Ket*. Cambridge University Press, 2005.

[12] L. Hashemi y B. Thomas, *Objective PET Self-study Pack (Student's Book with answers with CD-ROM and Audio CDs (3))*. Cambridge English, 2010.

[13] A. Capel y W. Sharp, *Objective first certificate: student's book*. Cambridge University Press, 2010.

[14] *Base de datos del Centro de Idiomas de la UTM*, 2021.

[15] E. Productions, *elllo*, visitado el 3-may-2023. dirección: `https://elllo.org/english/levels/index.htm`.

[16] R. Nogueira y K. Cho, "Task-oriented query reformulation with reinforcement learning", *arXiv preprint arXiv:1704.0457*, 2017.

[17] L. Hu y W. Yao, "Design and implementation of college English multimedia aided teaching resources", *The International Journal of Electrical Engineering & Education*, 2021.

[18] D. Ramírez Verdugo e I. Alonso Belmonte, "Using digital stories to improve listening comprehension with Spanish young learners of English", 2007.

[19] N. Guan, J. Song y D. Li, "On the advantages of computer multimedia-aided English teaching", *Procedia computer science*, vol. 131, págs. 727-732, 2018.

[20] F. M. Soares y A. M. Souza, *Neural network programming with Java*. Packt publishing Birmingham, UK, 2017.

[21] A. Serrano, E. Soria y J. Martin, "Redes neuronales artificiales", *Escuela Técnica Superior de Ingeniería, Curso de Doctorado*, págs. 15-19, 2009.

[22] W. R. Asanza y B. M. Olivo, "Redes neuronales artificiales aplicadas al reconocimiento de patrones", *Editorial UTMACH*, vol. 1, n.º 4, págs. 19-21, 2018.

[23] R. S. Sutton y A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[24] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau et al., "An introduction to deep reinforcement learning", *Foundations and Trends® in Machine Learning*, vol. 11, n.º 3-4, págs. 219-354, 2018.

[25] C. J. Watkins y P. Dayan, "Q-learning", *Machine learning*, vol. 8, n.º 3, págs. 279-292, 1992.

[26] B. C. Kuo, *Sistemas de control automático*. Pearson Educación, 1996.

[27] B. Rolf, I. Jackson, M. Müller, S. Lang, T. Reggelin y D. Ivanov, "A review on reinforcement learning algorithms and applications in supply chain management", *International Journal of Production Research*, vol. 61, n.º 20, págs. 7151-7179, 2023.

[28] A. Thomas, "Reinforcement Learning Cheatsheet", 2021.

[29] V. Mnih, A. P. Badia, M. Mirza et al., "Asynchronous methods for deep reinforcement learning", en *International conference on machine learning*, PMLR, 2016, págs. 1928-1937.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford y O. Klimov, "Proximal policy optimization algorithms", *arXiv preprint arXiv:1707.06347*, 2017.

[31] M. Lapan, "OpenAI Gym", en *Deep Reinforcement learning Hands-on*. Packt publishing, 2020, págs. 37-39.

[32] S. Baselines, *Stable Baselines docs*, visitado el 2-sep-2023. dirección: `https://stable-baselines.readthedocs.io/en/master/`.

[33] A. McCallumzy, K. Nigamy, J. Renniey y K. Seymorey, "Building domain-specific search engines with machine learning techniques", en *Proceedings of the AAAI Spring Symposium on Intelligent Agents in Cyberspace*, Citeseer, 1999, págs. 28-39.

[34] I. Arroyo-Fernández, C.-F. Méndez-Cruz, G. Sierra, J.-M. Torres-Moreno y G. Sidorov, "Unsupervised sentence representations as word information series: Revisiting TF–IDF", *Computer Speech & Language*, vol. 56, págs. 107-129, 2019.

[35] W. Nagy, J. Scott, M. Kamil, P. Mosenthal, P. Pearson y R. Barr, "Vocabulary processes", *Handbook of reading research*, vol. 3, 2000.

[36] D. M. Blei, A. Y. Ng y M. I. Jordan, "Latent dirichlet allocation", *Journal of machine Learning research*, vol. 3, n.º Jan, págs. 993-1022, 2003.

[37] L. Tool, *LanguageTool Tu corrector inteligente*, visitado el 5-ago-2023. dirección: `https://languagetool.org/es`.

[38] L. Driscoll, *Common Mistakes at KET: And How to Avoid Them* (Common Mistakes). Cambridge University Press, 2007, pág. 50, ISBN: 9780521692489.

[39] L. Driscoll, *Common mistakes at PET ... and how to avoid them* (Cambridge books for Cambridge exams). Cambridge University Press, 2005, pág. 18, ISBN: 9783125341272.

[40] mimno, *jsLDA: In-browser topic modeling*, visitado el 26-jul-2023. dirección: `https://mimno.infosci.cornell.edu/jsLDA`.