APRENDIZAJE POR REFUERZO DE UN PARSER SEMÁNTICO ÓPTIMO EN DRT

MAGISTER EN MATEMÁTICAS APLICADAS Y CIENCIAS DE LA COMPUTACIÓN

Jessenia Piza Londoño

Dirección:

Edgar José Andrade Lotero

Universidad del Rosario

Escuela de Ingeniería, Ciencia y Tecnología

Maestría en Matemáticas Aplicadas y Ciencias de la Computación

AGRADECIMIENTOS

Quiero expresar mi más profundo agradecimiento a mi director, Edgar Andrade, por ser mi guía y apoyo inquebrantable a lo largo de este viaje académico. Desde los primeros días de mi pregrado (e incluso ahora en la maestría), Edgar ha sido una inspiración constante, alentándome y cultivando mi interés en los temas de inteligencia artificial. Su pasión por diversos campos relacionados siempre ha despertado en mí un entusiasmo renovado y un deseo de esforzarme al máximo. Su orientación experta y dedicación incansable han sido fundamentales para mi crecimiento académico y profesional. Estoy profundamente agradecida por su inquebrantable apoyo y confianza en mí.

ABSTRACT

El procesamiento de lenguaje natural (NLP) se enfoca en crear sistemas de comunicación eficaces entre computadoras y humanos. El mayor avance en esta área se ha logrado mediante grandes modelos de lenguaje (LLM, por sus siglas en inglés) basados en representaciones vectoriales que implementan un tipo de memoria asociativa. No obstante, en dominios regidos por reglas, como relaciones espaciales o normas legales, la memoria asociativa tiene poca precisión y se necesita una representación más precisa. Para el procesamiento de este tipo de dominios se han utilizado parsers semánticos, los cuales asignan una representación lógica a los textos mediante un análisis de la estructura sintáctica y la interpretación semántica de sus componentes. Sin embargo, ellos están basados en reglas complejas y específicas, dificultando su diseño y mantenimiento. Además, como la implementación de estas reglas es manual, puede ser engorrosa y limitar su rendimiento. Este estudio propone un enfoque innovador mediante el uso de aprendizaje por refuerzo profundo para diseñar un parser semántico que pueda aprender y adaptarse de manera automática y eficiente. El agente aprenderá a asignar interpretaciones semánticas a través de recompensas y optimizará su comportamiento y rendimiento con el tiempo. La investigación se centrará en cuestiones clave, como la representación de reglas, las señales de recompensa y la evaluación de resultados. De esta manera, estos progresos pueden tener un impacto significativo en los avances tecnológicos de procesamiento de lenguaje natural.

Términos clave— representación formal del lenguaje, razonamiento automático, inferencia lógica, teoría de la representación del discurso, procesamiento del lenguaje natural.

Natural language processing (NLP) focuses on creating effective communication systems between computers and humans. The greatest progress in this area has been achieved through large language models (LLM) based on vector representations that implement a type of associative memory. However, in rule-governed domains, such as spatial relations or legal norms, associative memory has low precision and a more precise representation of it is needed. For the processing of this type of domains, semantic parsers have been used, which assign a logical representation to the texts through an analysis of the syntactic structure and the semantic interpretation of its components. However, they are based on complex and specific rules, making their design and maintenance difficult. Additionally, since implementation of these rules is manual, it can be cumbersome and limit your performance. This study proposes an innovative approach by using deep reinforcement learning to design a semantic parser that can learn and adapt automatically and efficiently. The agent will learn to assign semantic interpretations through rewards and optimize its behavior and performance over time. Research will focus on key questions such as rule representation, reward signals, and outcome evaluation. In this way, these progresses can have a significant impact on technological advances in natural language processing.

Index Terms— formal representation of language, automatic reasoning, logical inference, discourse representation theory, natural language processing.

Índice

Lista de figuras

Lista de tablas

1 INTRODUCCIÓN

El procesamiento de lenguaje natural (NLP, por sus siglas en inglés) es un campo que estudia las interacciones entre las computadoras y el lenguaje humano, capaz de diseñar mecanismos para comunicarse que sean eficaces computacionalmente [1, 2]. En este campo, un parser semántico es un componente que se encarga de asignar una representación semántica a un texto dado.

A medida que el procesamiento de lenguaje natural ha avanzado, se han producido notables progresos, especialmente en la representación y comprensión del lenguaje. Actualmente, los modelos de vanguardia se basan en el uso de grandes modelos de lenguaje, donde las frases se interpretan como vectores y la comprensión del texto se asemeja a una memoria asociativa, influenciada por la estructura estadística de los textos [3]. Sin embargo, hay dominios que son altamente regidos por reglas (relaciones espaciales, normas morales y/o legales, etc) cuya comprensión requiere más que una asociación estadística y que se puede beneficiar de una representación lógica.

El análisis del parsing semántico se puede utilizar para diferentes aplicaciones que necesiten la comprensión más profunda del escrito. A modo de ilustración, considere la oración "El libro está encima del escritorio que está al lado izquierdo de la silla", donde quisiéramos poder responder preguntas como: "¿qué está debajo del libro?", "¿qué está al lado derecho del escritorio?", entre otros.

Tradicionalmente, los parsers semánticos son sistemas que se diseñan utilizando reglas gramaticales y semánticas que son definidas por humanos. Sin embargo, esto tiene varias limitaciones [2]. Principalmente, las reglas gramaticales y semánticas pueden ser complejas y demasiado específicas, lo que puede dificultar su diseño y mantenimiento. Así mismo, estas reglas pueden ser engorrosas de implementar manualmente, lo que ralentiza el rendimiento de estos. De esta manera, surge la pregunta de si existe la posibilidad de que la máquina pueda aprender ella misma estas reglas de parsing cuyo funcionamiento sea incluso más óptimo que las reglas diseñadas manualmente.

El propósito de esta investigación y su objeto de estudio es el diseño de parsers semánticos que sea más eficiente y flexible que los enfoques convencionales. De esta manera, se propone utilizar técnicas de aprendizaje por refuerzo profundo para diseñar un parser semántico, donde, el agente aprenderá a asignar significados semánticos a cada elemento del texto, y, mediante una señal de recompensa, cree sus propias reglas

para optimizar su desempeño.

La pregunta surge naturalmente: ¿Puede entrenarse un agente para ejecutar las reglas apropiadas en cada etapa de la representación parcial únicamente observando qué estructura final es adecuada para una tarea de NLP dada? Al adoptar este enfoque, efectivamente delegamos todas las complejidades lingüísticas al agente, quien, a través de ensayo y error iterativo, aprende autónomamente las reglas necesarias.

Presentamos los hallazgos preliminares de un agente de aprendizaje por refuerzo profundo entrenado en un entorno donde los estados corresponden a representaciones semánticas parciales de oraciones simples. Específicamente, los estados son tuplas que contienen una oración (enmascarada) y una Estructura de Representación del Discurso (DRS), como lo define la Teoría de Representación del Discurso [4]. Las acciones dentro de este entorno implican modificaciones a la oración procesada (enmascarando una palabra específica) y adiciones a la DRS (añadiendo referentes y condiciones). La señal de recompensa está diseñada para penalizar acciones innecesarias y DRS finales inapropiadas que no respondan a una pregunta básica sobre la oración. Nuestros hallazgos demuestran que el agente aprende con éxito a generar una DRS adecuada no sólo para la pregunta de entrenamiento, sino también para consultas relacionadas. El trabajo futuro se centrará en mejorar el entorno para permitir la representación de un espectro más amplio de oraciones en DRS.

2 OBJETIVOS

2.1. Objetivo general

Diseñar, implementar y evaluar un parser semántico basado en aprendizaje por refuerzo, demostrando su eficiencia y flexibilidad.

2.2. Objetivos específicos

1. Representar las reglas de un parser semántico como un problema de aprendizaje por refuerzo.
2. Implementar uno de los algoritmos de aprendizaje por refuerzo vistos con detalle en el curso de la maestría, para entrenar un parser semántico.
3. Evaluar el rendimiento del parser semántico desarrollado.

3 PROBLEMA Y JUSTIFICACIÓN

La representación de la información transmitida a través del lenguaje natural plantea un desafío fundamental en los ámbitos de la Inteligencia Artificial y la Ciencia Cognitiva. Este desafío implica la conversión de "datos no estructurados" (es decir, secuencias de caracteres) en representaciones computacionalmente tratables. Estas representaciones no deben limitarse a procesar caracteres individuales o próximos de forma aislada; sino que deben ser capaces de realizar un espectro de tareas de Procesamiento del Lenguaje Natural (NLP), incluyendo inferencia, respuesta a preguntas, traducción y más [2]. Durante décadas, los enfoques predominantes se basaron en diversos formalismos lógicos para lograr este objetivo [2,5]. Sin embargo, el panorama cambió drásticamente con la aparición de las redes neuronales profundas, particularmente la arquitectura del transformer [6], que desde entonces ha cobrado gran relevancia.

Las redes neuronales han demostrado ser altamente eficaces para diversas tareas de procesamiento de lenguaje natural gracias a su capacidad de codificar información en vectores, conocidos como *embeddings*, que configuran un espacio semántico [2]. Sin embargo, estos vectores, aunque útiles para tareas computacionales, no están diseñados intrínsecamente para realizar inferencias, una habilidad en la que los modelos de transformers a menudo fallan [7–9]. La limitación en estos modelos llega a ser crítica en aplicaciones que requieren procesos de inferencia confiables y explicables, donde los formalismos lógicos tradicionales aún ofrecen una alternativa sólida.

No obstante, la comprensión e implementación de los formalismos lógicos tradicionales son tareas a menudo laboriosas [10, 11]. Además, los procesos de transformación automática involucrados en convertir oraciones de lenguaje natural en representaciones lógicas añaden capas adicionales de complejidad y sofisticación [5, 12, 13]. Dado este panorama, surge la necesidad de explorar si es posible entrenar a un agente de aprendizaje profundo por refuerzo para que pueda ejecutar las reglas necesarias en cada etapa de la representación parcial del lenguaje basado en una tarea específica de NLP. Este enfoque podría simplificar significativamente el proceso de conversión y permitir una adopción más amplia y efectiva de representaciones lógicas en sistemas de inteligencia artificial, integrando la flexibilidad del lenguaje natural con la precisión de las inferencias lógicas.

3.1. Aplicaciones

3.1.1. Creación de ontologías

Las ontologías son representaciones estructuradas del conocimiento en un dominio específico. Ahora bien, uno de los objetivos finales del Procesamiento de Lenguaje Natural es la lectura automática, es decir, la comprensión no supervisada del texto. Esto se busca a través del análisis semántico, que convierte el texto en una representación de significado. Sin embargo, para que esta representación sea útil, debe estar predefinida y estructurada para apoyar el razonamiento, y las ontologías cumplen esta función al proporcionar un vocabulario común y apoyar el razonamiento necesario para entender el texto [14]. De esta manera, tomando como base la representación de significado creada por el parser semántico, es capaz de crear una ontología capaz de razonar y responder problemas.

3.1.2. Conjunto de datos de respuesta a preguntas (QA) de tipo "multi-hop"

Un conjunto de datos de respuesta a preguntas (QA) de tipo multi-hop tiene como objetivo evaluar las habilidades de razonamiento e inferencia al requerir que un modelo lea varios párrafos para responder a una pregunta determinada [15]. Algunos de estos conjuntos son HotpotQA y StrategyQA.

HotpotQA presenta un desafío significativo al requerir que los sistemas de QA razonen sobre múltiples documentos de apoyo para proporcionar respuestas a preguntas complejas. Las preguntas en este conjunto de datos no están limitadas a esquemas de conocimiento preexistentes, y se proporcionan hechos de apoyo a nivel de oración que facilitan el razonamiento explicativo. Además, el conjunto incluye preguntas que requieren la comparación de hechos [16]. En este contexto, un parser semántico puede descomponer las preguntas en componentes semánticos clave, tales como entidades y relaciones, permitiendo una identificación precisa de la información relevante en varios documentos. Además, puede estructurar las respuestas de manera que refleje claramente el proceso de razonamiento subyacente, alineándose con el objetivo de HotpotQA de proporcionar explicaciones detalladas.

Por otro lado, StrategyQA introduce un nuevo desafío al requerir que los sistemas de QA infieran pasos de razonamiento implícitos en las preguntas, en lugar de seguir

una secuencia explícita. El conjunto de datos abarca una amplia gama de estrategias de razonamiento y proporciona anotaciones detalladas que incluyen la descomposición en pasos de razonamiento y párrafos de evidencia [17]. En este escenario, un parser semántico facilita la descomposición de preguntas en componentes semánticos y la correspondencia de estos con los párrafos de evidencia relevantes, apoyando la extracción y organización de la información necesaria para cada paso del razonamiento.

4 MARCO TEÓRICO Y ESTADO DEL ARTE

4.1. Procesamiento de Lenguaje Natural

El procesamiento del lenguaje natural es el conjunto de métodos para hacer que el lenguaje humano sea accesible a las computadoras. En la última década, el procesamiento del lenguaje natural se ha integrado en nuestra vida diaria: la traducción automática es omnipresente en la web y en las redes sociales; la clasificación de texto evita que los correos electrónicos colapsen bajo una avalancha de spam; los motores de búsqueda han ido más allá de la coincidencia de cadenas y el análisis de redes hacia un alto grado de sofisticación lingüística; los sistemas de diálogo proporcionan una forma cada vez más común y eficaz de obtener y compartir información [1].

4.2. Relaciones lógicas en el lenguaje

La construcción de sistemas que construyen la sintaxis del lenguaje natural mediante etiquetado y análisis son importantes y necesarias para la resolución de algunos problemas sobre el procesamiento de lenguaje natural. Sin embargo, para aplicaciones más potenciales y prometedores es necesario ir más allá de la sintaxis a la semántica: el significado subyacente del texto. De esta manera, poder resolver problemas como la solución de preguntas en un texto [1, 2].

El análisis semántico implica convertir el lenguaje natural en una representación basada en el significado. Uno de los criterios para una representación de significado es que los enunciados en la representación no deben ser ambiguos, es decir, deben tener sólo una interpretación posible. Ahora bien, el lenguaje natural no tiene esta propiedad, puede tener múltiples interpretaciones.

En el contexto de un problema de parsing semántico, consideremos la formalización de relaciones familiares como un ejemplo ilustrativo. Definimos **constantes** para representar individuos, como "Juan" y "María". Introducimos **relaciones**, como "PadreDe" y "MadreDe", para expresar las conexiones familiares. Establecemos un **modelo** que asigna a estas relaciones un significado específico, por ejemplo, "PadreDe(Juan, María)" indica que Juan es el padre de María. Además, si incorporamos las relaciones fraternales como "EsHermanoDe" y afirmamos que "EsPadreDe(Juan, María)" y "EsHermanoDe(María, Ana)", se puede deducir lógicamente que Juan es padre de Ana (tomando un contexto donde no pueden ser

hermanastras). A este enfoque del significado se le conoce como **semántica teórica de modelos**, el cual facilita la representación y comprensión de la estructura semántica del problema, eliminando las ambigüedades que se puedan presentar y permitiendo un análisis más claro y preciso en el proceso de parsing.

4.2.1. Lógica de primer orden

La lógica de primer orden (o FOL por sus siglas en inglés) es un lenguaje de representación de significados, flexible, bien comprendido y manejable computacionalmente. Esta proporciona una base computacional sólida para los requisitos de verificabilidad, inferencia y expresividad, así como una semántica teórica de modelos sólida [12]. Los elementos atómicos de FOL son los siguientes:

- **Término:** Es el dispositivo capaz de representar los objetos.
- **Constantes:** Son los objetos específicos del mundo que se describe. Estas se representan con letras mayúsculas o con nombres propios como María. Las constantes se refieren a exactamente un objeto, pero, los objetos pueden tener varias constantes que se refieran a ellos.
- **Variables:** Nos permiten hacer afirmaciones sobre objetos sin tener que hacer referencia a ningún objeto en particular. Estas se representan con minúsculas.
- **Predicados:** Se refieren a los símbolos que nombran las relaciones que se mantienen entre un número fijo de objetos en un dominio determinado.

Tomando estos elementos, podemos representar la oración "Pedro mira a María" como MIRAR(PEDRO, MARÍA).

Ahora bien, otro elemento muy útil que se le puede agregar a FOL es la **notación lambda** [12]. Esta proporciona una forma de abstracción de las fórmulas FOL completamente específicas de un modo que resulta útil para el análisis semántico y que tiene la siguiente forma:

$$\lambda x.P(x),$$

que consisten en el símbolo griego λ, seguido de una o más variables, seguida de una fórmula FOL que utiliza esas variables. Así, la utilidad de estas expresiones λ se basa en la posibilidad de aplicarlas a términos lógicos para producir nuevas expresiones FOL

donde las variables de los parámetros formales están ligadas a los términos especificados. Este proceso se conoce como reducción λ, y consiste en una simple sustitución textual de las variables λ y la eliminación del λ [2]. Para una mejor ilustración consideremos la expresión λ más compleja:

$$\lambda x.\lambda y.\text{CERCA}(x, y)$$

Aplicamos esta expresión al término "Parque":

$$\lambda x.\lambda y.\text{CERCA}(x, y)(\text{PARQUE})$$

$$\lambda y.\text{CERCA}(\text{PARQUE}, y)$$

Podemos aplicar la nueva expresión λ resultante al término "Café":

$$\lambda y.\text{CERCA}(\text{PARQUE}, y)(\text{CAFÉ})$$

$$\text{CERCA}(\text{PARQUE}, \text{CAFÉ})$$

Con esto, se puede convertir un predicado con múltiples argumentos en una secuencia de un solo argumento [1].

La lógica de primer orden se instituye como el formalismo lógico de preferencia y versátil empleado para la representación del conocimiento [5]. El enfoque de esta investigación se centra específicamente en la representación de eventos dentro de este marco [4, 18, 19]. Los eventos nacen como entidades fundamentales junto con las personas y objetos, sujetos a modificaciones e interacción con otras entidades a través de los predicados. Aprovechar estos eventos como elementos primitivos facilita la derivación de numerosas inferencias sobre situaciones, resolviendo problemas derivados de la estructura de argumentos flexible propio del lenguaje natural. Los siguientes ejemplos ilustran este concepto:

(1) Pedro comió.

(2) Pedro comió un sandwich.

(3) Pedro comió un sandwich a la 1pm.

El evento que denota a una persona comiendo puede ser simbolizado utilizando una constante de evento, designada como e, la cual es susceptible a modificaciones flexibles

a través de predicados. En consecuencia, las oraciones anteriores (1)–(3) pueden ser codificadas en las fórmulas correspondientes (4)–(6), de la siguiente manera:

(4) $\textsc{comer}(e) \wedge \textsc{sujeto}(e, p)$

(5) $\textsc{comer}(e) \wedge \textsc{sujeto}(e, p) \wedge \textsc{objeto}(e, s)$

(6) $\textsc{comer}(e) \wedge \textsc{subject}(e, p) \wedge \textsc{objeto}(e, s) \wedge \textsc{sucede}(e, 1pm)$

Dentro de un marco temporal discreto y un universo discreto e identificable, el cuantificador de existencia puede ser sustituido por disyunciones, mientras que el cuantificador universal puede ser reemplazado por conjunciones. Esta adaptación tiene como objetivo depender únicamente de fórmulas fundamentadas, que equivalen a proposiciones atómicas, permitiendo así una inferencia eficiente mediante solucionadores de proposiciones SAT (SAT-solvers).

4.2.2. Solucionadores SAT

Los solucionadores SAT abarcan algoritmos eficientes destinados a determinar si una fórmula de lógica proposicional tiene un modelo [5]. Este proceso implica establecer si existe una combinación de valores de verdad que puedan ser asignados a sus componentes atómicos de tal manera que toda la fórmula se evalúe como verdadera. Los problemas de inferencia, donde el objetivo es determinar si una fórmula ψ se sigue lógicamente de un conjunto de fórmulas $\Gamma = \{A_1, \ldots, A_n\}$, pueden reformularse como la determinación de la satisfiabilidad de la fórmula $\neg\big(\bigwedge_{i=1}^{n} A_i \rightarrow \psi\big)$. En la Figura 1, mostramos cómo esta puede aplicarse para abordar consultas relacionadas con afirmaciones fácticas en un contexto específico.

Cabe destacar que el ejemplo proporcionado en la Figura 1 es intencionalmente simple para mayor claridad en la presentación. Surgen escenarios más complejos cuando, por ejemplo, la consulta se altera a $\lambda x \textsc{derecha}(\textsc{x},\textsc{B})$. En tales casos, lograr una solución satisfactoria requiere conocimiento común sobre las relaciones espaciales entre objetos y cómo estas relaciones se propagan dentro del conjunto. Este tipo de conocimiento común puede representarse como fórmulas fundamentadas de lógica de predicados (por ejemplo, $\bigwedge_{x,y \in \text{objetos}}\big(\textsc{izquierda}(x, y) \rightarrow \textsc{derecha}(y, x)\big)\big)$.

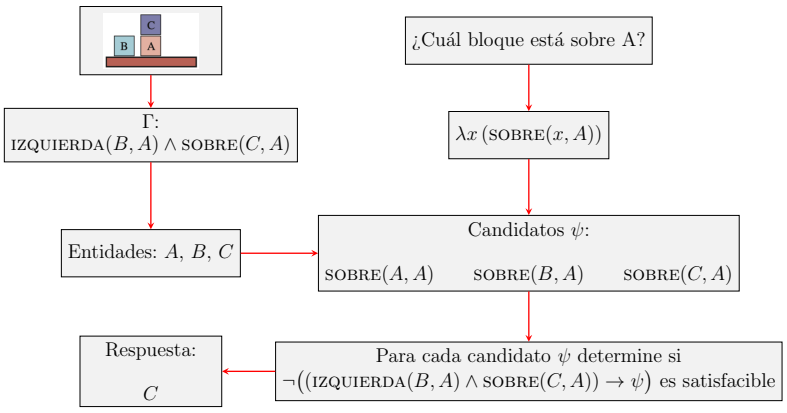

Figura 1: Resolución de preguntas mediante SAT-solvers. El proceso de responder preguntas fácticas utilizando fórmulas fundamentadas de lógica de predicados y un SAT-solver. Necesitamos representar el escenario usando fórmulas para expresar conocimiento de hechos y reglas de sentido común (rectángulo superior izquierdo y el inferior). A partir de estas fórmulas extraemos las entidades mencionadas. Después, representamos la pregunta como una fórmula funcional en el cálculo lambda, lo que genera candidatos al aplicar cada entidad a esta fórmula. Utilizamos el SAT-solver para determinar si la representación del escenario implica cada candidato, y reportamos las entidades que producen candidatos implicados por el escenario. Fuente: Elaboración propia.

4.3. Teoría de representación de discursos

La representación de oraciones del lenguaje natural mediante fórmulas lógicas es el ámbito de la semántica formal [12, 20].Este método de representación fue extendido al tratamiento del discurso mediante la Teoría de Representación de Discursos [4], o DRT por sus siglas en inglés. Esta teoría parte de la idea de que el discurso se genera oración por oración, las cuales se van integrando en una representación semántica unificada. El concepto básico aquí es una estructura de representación de discursos (o DRS por sus siglas en inglés).

En la Figura 2b, se puede observar cómo las DRS pueden anidarse permitiendo el tratamiento de cierto tipo de anáforas, como la anáfora de acotación. Por ejemplo, el pronombre anafórico "sus" en la oración "Toda bailarina sufre en sus pies" está ligado a la referencia "bailarina". Este tipo de dependencia del pronombre se representa adecuadamente mediante la subDRS condicional que se observa en la Figura 2b. En esta representación, el referente y sólo puede ser accesible por las condiciones dentro de la caja $b3$ y no por ninguna otra, de tal manera que la variable y queda acotada por el alcance de x, que representa a la bailarina (vea los detalles en [4], cap. 2).

Otra ventaja de emplear DRT en la representación semántica es su facilidad para la

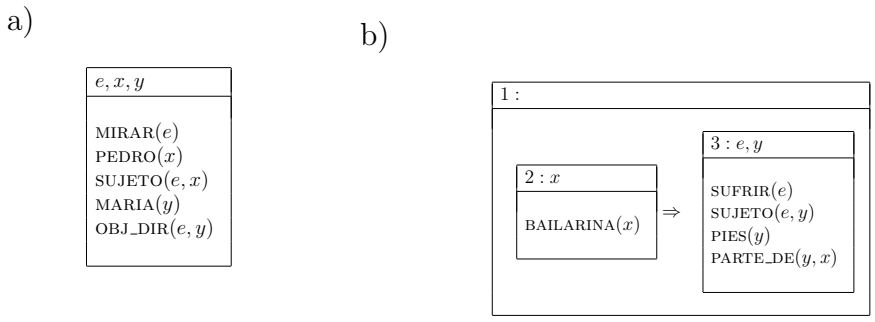

Figura 2: **Ejemplos de estructuras de representación de discursos.** (a) DRS apropiada para la oración "Pedro mira a María". Esta se compone de una colección de referentes (caja superior) y una colección de condiciones que cumplen estos referentes (caja inferior). (b) DRS apropiada para la oración "Toda bailarina sufre en sus pies". En la colección de condiciones se incluyen dos DRS relacionadas por un condicional. Fuente: Elaboración propia.

integración entre varias DRS. Por ejemplo, las DRS que representan dos oraciones separadas (como en la interpretación del discurso oración por oración) tienen un proceso de integración bastante sencillo, que también se aplica a representaciones parciales de diferentes fragmentos dentro de la misma oración. Además, las DRS son útiles para ampliar las representaciones e incluir tiempo verbal y aspecto verbal, actitudes proposicionales y presuposiciones [21]. Por último, cada DRS es equivalente a una fórmula de lógica de predicados. Por ejemplo, las dos DRS en la Figura 2 producen las fórmulas (7) y (8):

(7) $\exists x, y, e\big(\text{MIRAR}(e) \wedge \text{PEDRO}(x) \wedge \text{SUJETO}(e, x) \wedge \text{MARÍA}(y) \wedge \text{OBJ\_DIR}(e, y)\big)$

(8) $\forall x\Big(\text{BAILARINA}(x) \rightarrow \exists y, e\big(\text{SUFRIR}(e) \wedge \text{PIES}(y) \wedge \text{PARTE\_DE}(y, x) \wedge \text{SUJETO}(e, y)\big)\Big)$

4.4. Resolución de preguntas

El proceso de resolución de preguntas utilizando lógica de primer orden y el cálculo lambda consta de varios pasos. En primer lugar, se transforma el texto en una DRS lo cual implica asignar símbolos y conectivos lógicos a las diferentes palabras y estructuras gramaticales del texto. A su vez, esta DRS puede representarse mediante una fórmula de la lógica de primer orden. Posteriormente, se identifican las constantes presentes en el texto, tales como individuos, propiedades o eventos, para utilizarlas después en el proceso de deducción. Luego, se transforma la pregunta en una fórmula del cálculo lambda con una sola variable libre. A partir de esta fórmula, se crea una lista de fórmulas de la lógica de primer orden, reemplazando la variable libre por cada una de las constantes de tipo apropiado identificadas previamente. Estas fórmulas son los posibles

candidatos de respuestas. Por último, se intenta deducir lógicamente cada candidato a partir de la fórmula obtenida en el primer paso. Si alguna deducción es válida, entonces el candidato se considera como la respuesta a la pregunta planteada en el texto original. Dado que se tienen múltiples candidatos, es posible que más de uno de los candidatos responda la pregunta. En este caso, el primero de ellos que sea encontrado durante la iteración sería determinado como la respuesta [5].

4.5. Aprendizaje por refuerzo

El aprendizaje por refuerzo (o RL por sus siglas en inglés) consiste en aprender qué hacer para maximizar una señal de recompensa numérica. Un agente debe realizar diferentes acciones en un entorno determinado para descubrir cuáles de estas son las que generan una recompensa mayor. En algunos casos, las acciones pueden afectar no sólo la recompensa inmediata, sino también la de la siguiente iteración y, a través de ella, todas las posteriores. Estas dos características (búsqueda de prueba-error y recompensa retrasada) son las dos más distintivas en el aprendizaje por refuerzo [22].

Además, más allá del agente y del entorno, se pueden identificar cuatro subelementos principales en el sistema de aprendizaje por refuerzo. Estos son:

- **Política:** Define el comportamiento del agente en un estado dado.
- **Señal de recompensa:** Define el objetivo del problema. Indica si es bueno en un tiempo inmediato.
- **Función de valor:** Indica la señal de recompensa a largo plazo.
- **Modelo (opcional):** Imita el comportamiento del entorno y se logran dar inferencias de este. Por ejemplo, dado un estado y una acción, el modelo proporciona la distribución de probabilidad del siguiente estado y su respectiva recompensa.

La categoría predominante de políticas estocásticas se conoce como estrategia ϵ-greedy. En esta estrategia, cuando se enfrenta a un estado particular, el agente selecciona la acción que maximiza la recompensa a largo plazo (según las estimaciones actuales) con una probabilidad de $1 - \epsilon$, mientras que opta por una de las acciones alternativas con una probabilidad de ϵ.

El objetivo del aprendizaje por refuerzo es identificar la política óptima, que maximiza el valor en todos los estados. Sin embargo, en entornos con un gran número de

estados, integrar el aprendizaje por refuerzo con redes neuronales profundas se convierte en una alternativa viable [23, 24]. Estas redes tienen la capacidad de extraer las características de los estados que están relacionadas con las recompensas esperadas específicas, reduciendo virtualmente la complejidad del entorno y mejorando el proceso de aprendizaje.

4.6. Método de Entrenamiento de un Agente de Aprendizaje por Refuerzo utilizando Deep Q-Networks

Deep Q-Networks (DQN) es un enfoque popular que combina Q-learning con redes neuronales profundas para manejar espacios de estado y acción continuos. Este utiliza una red neuronal para aproximar la función de valor de acción, $Q(s, a)$, que estima la recompensa esperada de tomar una acción a en un estado s. La estructura básica del entrenamiento con DQN incluye los siguientes pasos [22, 25]:

1. **Inicialización:** Inicializar una red Q con pesos aleatorios y una red objetivo, que es una copia de la red Q.

2. **Interacción con el entorno:** En cada paso de tiempo t, el agente observa el estado s_t y selecciona una acción a_t utilizando una política ϵ-greedy.

3. **Recolección de experiencias:** Ejecutar la acción a_t en el entorno, observar la recompensa r_t y el siguiente estado s_{t+1}. Almacenar la experiencia (s_t, a_t, r_t, s_{t+1}) en un buffer de memoria de repetición.

4. **Actualización de la red Q:** Periódicamente, muestrear un minibatch de experiencias aleatorias desde el buffer de memoria. Para cada experiencia, calcular la actualización de la red Q utilizando la ecuación de Bellman:

$$y_j = r_j + \gamma \max_{a'} Q'(s_{j+1}, a'; \theta^-)$$

   donde γ es el factor de descuento, Q' es la red objetivo y θ^- son los parámetros de la red objetivo.

5. **Optimización:** Actualizar los parámetros de la red Q minimizando el error cuadrático medio entre y_j y $Q(s_j, a_j; \theta)$.

6. **Actualización de la red objetivo:** Cada cierto número de pasos, copiar los parámetros de la red Q a la red objetivo:

$$\theta^- \leftarrow \theta$$

El proceso de entrenamiento continúa hasta que el agente logra un rendimiento satisfactorio. DQN ha sido utilizado con éxito en diversas tareas de control y juegos, demostrando su capacidad para aprender políticas efectivas en entornos complejos y de alta dimensionalidad [25].

Además de las DQN, existen otros métodos avanzados en el campo del aprendizaje por refuerzo que son adecuados para entornos continuos y discretos. Uno de los algoritmos más destacados es el Proximal Policy Optimization (PPO). Esta es una técnica de optimización de políticas basada en gradientes que mejora la estabilidad y eficiencia en el aprendizaje, corrigiendo las deficiencias de algoritmos anteriores como el Trust Region Policy Optimization (TRPO). PPO utiliza una metodología de actualización que mantiene las actualizaciones de políticas dentro de una región de confianza, lo que evita cambios drásticos y garantiza un aprendizaje más estable [26].

Proximal Policy Optimization es particularmente útil en entornos con espacios de estado y acción continuos, donde las acciones pueden variar en un rango continuo en lugar de estar limitadas a un conjunto discreto. Este método ha demostrado ser eficaz en una variedad de aplicaciones complejas, como el control robótico y los videojuegos, debido a su capacidad para manejar la alta dimensionalidad y la variabilidad de las acciones [26].

Por otro lado, DQN es especialmente adecuado para entornos con espacios de estado continuos y acciones discretas. Para el propósito de este ejercicio, nos enfocaremos en el uso de DQN debido a que las acciones del entorno específico con el que trabajaremos son discretas. Además, nuestro propósito era darle continuidad a los métodos vistos en el curso y este fue el algoritmo que ahondamos.

4.7. Uso de RL en NLP

El aprendizaje por refuerzo (RL) ha demostrado ser una herramienta valiosa en la mejora de diversos sistemas de procesamiento de lenguaje natural, incluyendo la traducción automática, donde permite a los modelos aprender de manera más efectiva a través

de retroalimentación directa, optimizando no solo la probabilidad de las secuencias de palabras, sino también la calidad de la traducción basada en métricas específicas. En la tarea de resumir textos, el RL ha sido clave para desarrollar modelos que generan resúmenes coherentes y concisos, refinando su capacidad para captar y presentar la información más relevante. Asimismo, los sistemas de generación de lenguaje natural han visto grandes mejoras con la integración de RL, produciendo textos gramaticalmente correctos y contextualmente apropiados mediante la optimización de la fluidez y coherencia del lenguaje generado a través de recompensas basadas en la calidad del texto. Esto es especialmente útil en aplicaciones como la redacción automática de informes y la generación de contenido creativo. En el desarrollo de chatbots y asistentes virtuales, el RL ha permitido la creación de sistemas más interactivos y eficientes, capaces de mantener conversaciones más naturales y útiles, adaptándose mejor a las necesidades y respuestas de los usuarios gracias a la retroalimentación continua, resultando en una experiencia de usuario más satisfactoria y efectiva. Aunque es menos común, el RL también ha mejorado tareas de entendimiento del lenguaje, como la desambiguación de sentidos de las palabras y la resolución de referencias, al recibir recompensas por la correcta interpretación y respuesta a las consultas, permitiendo a los modelos mejorar su precisión en estas tareas. Estos avances han sido posibles gracias a la capacidad del RL para aprender de manera iterativa y adaptativa, optimizando continuamente sus estrategias basadas en la retroalimentación recibida. [27]

4.8. Trabajo relacionado

4.8.1. Parsers semánticos

Las reglas de construcción de una DRS relacionan una porción del árbol de análisis sintáctico de la oración con un elemento específico que debe incluirse en la DRS. La formulación gramatical original de estas reglas es una Gramática Libre de Contexto (CFG, por sus siglas en inglés) [4], aunque también existen versiones basadas en la Gramática Categorial Combinatoria [28]. Recientemente, el proceso de análisis sintáctico para utilizar DRT como representación semántica se ha llevado a cabo utilizando una red neuronal artificial de sequence-to-sequence, con resultados que superan la alternativa basada en reglas [29]. Las distintas variaciones de esta red fueron entrenadas con los datos del Parallel Meaning Bank (PAMEBA), logrando rendimientos por encima de

modelos tradicionales como el parsing semántico de Boxer [28].

Sin embargo, esta metodología depende de cómo se etiquetaron las oraciones. En este caso, el análisis sintáctico neuronal en cuestión se entrenó con PAMEBA, que etiqueta las oraciones como DRS con una forma específica. Si bien esta forma es exhaustiva y robusta, no es necesariamente universal. Al revisar varios ejemplos de datos etiquetados en este conjunto de datos, encontramos características que no son necesariamente óptimas. Se puede argumentar que el tratamiento de los plurales es discutible; además, a menudo aparecen representadas presuposiciones de las oraciones, aunque esto supone una carga computacional innecesaria en muchos casos de aplicación. Por último, analizar anáforas temporales y aspectos del tiempo verbal requiere distinciones muy finas [19], cuyo tratamiento exige representaciones semánticas detalladas más allá del tratamiento tradicional del DRT [30]. Nuestra metodología asume la libertad de incluir el tipo de detalles formales considerados apropiados para la tarea en cuestión, permitiendo modificaciones al aplicar dicha representación a otros dominios. Esta libertad no la proporciona el análisis sintáctico neuronal basado en aprendizaje supervisado.

4.8.2. Transformers

Los LLMs son hábiles en la interpretación de indicaciones en lenguaje natural y en la generación de texto mediante el aprovechamiento de patrones estadísticos arraigados en los datos de entrenamiento. Es especialmente notable la arquitectura basada en transformers, particularmente aquellas dotadas de conjuntos de parámetros extensos, que han demostrado una notable capacidad de razonamiento [31]. La ampliación de estos LLMs con técnicas de Generación Aumentada por Recuperación (RAG, por sus siglas en ingles) facilita la producción de respuestas fundamentadas en su precisión. Esto implica emplear transformers auto-codificadores para codificar segmentos de texto en vectores densos, que luego se agregan en un repositorio de vectores. Algoritmos de búsqueda eficientes recuperan posteriormente los textos más pertinentes similares a una consulta, que también ha sido codificada como un vector. Estos segmentos de texto se introducen luego en la indicación presentada al LLM, equipándolo así con conocimiento factual para informar la generación de su respuesta. Sin embargo, a pesar del considerable reconocimiento que rodea a los LLMs, numerosos estudios han revelado la inherente falta de fiabilidad de sus capacidades de razonamiento [7–9].

Arquitectura del transformer

La arquitectura del transformer tiene una estructura de encoder-decoder. Aquí, la arquitectura general consiste en que el encoder mapea una secuencia de entrada de representaciones de símbolos $(x_1, \ldots, x_n)$ a una secuencia de representaciones continúas $z = (z_1, \ldots, z_n)$. Dado z, el decoder genera una secuencia de salida $y_1, \ldots, y_m$ símbolos un elemento a la vez. En cada paso, el modelo es auto-regresivo, consumiendo los símbolos previamente generados como entrada adicional al generar el siguiente. Así, el transformer sigue esto utilizando capas apiladas de auto-atención y capas completamente conectadas punto a punto tanto para el encoder como para el decoder, como se muestra en las mitades izquierda y derecha de la Figura 3, respectivamente. [6].

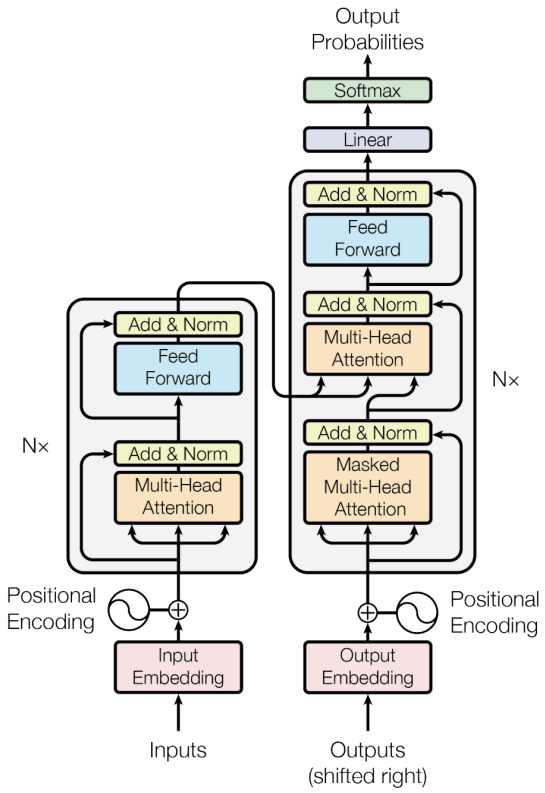

Figura 3: Arquitectura del transformer

- **Encoder:** Es responsable de procesar la secuencia de entrada y generar una representación codificada. Está compuesto por una pila de capas idénticas, cada una de las cuales contiene dos subcapas principales: una capa de autoatención y una red neuronal feed-forward.
- **Decoder:** Tiene una estructura similar al encoder pero con una capa adicional de

atención cruzada. Está compuesto por una pila de capas idénticas, cada una de las cuales contiene tres subcapas principales: una capa de autoatención, una capa de atención cruzada y una red neuronal feed-forward.

- **Atención:** Una función de atención puede describirse como el mapeo de una consulta y un conjunto de pares clave-valor a una salida, donde la consulta, las claves, los valores y la salida son todos vectores. La salida se calcula como una suma ponderada de los valores, donde el peso asignado a cada valor se calcula mediante una función de compatibilidad de la consulta con la clave correspondiente.

Ahora bien, los modelos de solo encoder (encoder-only) como BERT están diseñados para tareas de comprensión del lenguaje, aprovechando el contexto bidireccional de la secuencia de entrada para realizar tareas como clasificación y reconocimiento de entidades. Los modelos de solo decoder (decoder-only) como GPT, en cambio, se enfocan en la generación de texto, utilizando un contexto unidireccional para predecir la siguiente palabra en una secuencia dada. Finalmente, los modelos encoder-decoder, como el transformer original, combinan ambas partes de la arquitectura para tareas de transformación de secuencias, permitiendo la codificación de la entrada a través del encoder y la generación de la salida mediante el decoder, siendo ideales para aplicaciones como la traducción automática y el resumen de texto. [6].

Dado el objetivo de esta investigación, que requiere la comprensión del lenguaje para la creación de un parser semántico y debido a que este sólo necesita los embeddings para interpretar los estados crudos (lo cual se detallará más adelante en la sección 6.1.1), se utilizará un modelo encoder-only.

5 METODOLOGÍA

Para llevar a cabo la metodología de la investigación, se utilizará Python como software principal de toda la investigación. Además, la metodología que se adelantará en este trabajo, se basará en el ciclo de desarrollo del modelo como se muestra en la Figura 4.

1. **Diseño del Modelo DRL (Deep Reinforcement Learning):**

   a) Creación de un entorno de aprendizaje reforzado tradicional con el diseño de la función de recompensa, definición de acciones permitidas y la política respectiva.

   b) Selección de la arquitectura de red neuronal para el modelo DRL.

   c) Ajuste del entorno de aprendizaje para el DRL.

2. **Experimentación:**

   a) Adquisición de datos de entrenamiento, validación y prueba. Estos serían de la siguiente manera:

   - **Conjunto de entrenamiento:** El agente interactúa con el entorno y aprende a través de estas interacciones. Es fundamental para que el agente adquiera políticas de acción óptimas.
   - **Conjunto de validación:** Este conjunto se utiliza para ajustar hiperparámetros del modelo, como la tasa de aprendizaje o la arquitectura de la red neuronal. No se utiliza directamente en el proceso de entrenamiento, pero ayuda a evitar el sobreajuste.
   - **Conjunto de prueba:** El agente interactúa con el entorno, y los resultados se utilizan para medir la calidad de la política aprendida.

   b) Entrenamiento del modelo DRL utilizando el conjunto de entrenamiento.

   c) Evaluación del modelo.

   d) Optimización de hiperparámetros respectivos y ajuste del modelo en función de los resultados.

3. **Evaluación de resultados:**

a) Evaluación del modelo con el conjunto de prueba utilizando métricas de rendimiento específicas para tareas de análisis semántico.

b) Análisis cualitativo para comprender las causas de los errores.

4. **Mejora del modelo:** Iteración sobre el diseño del modelo y ajustes basados en los resultados del análisis de errores.

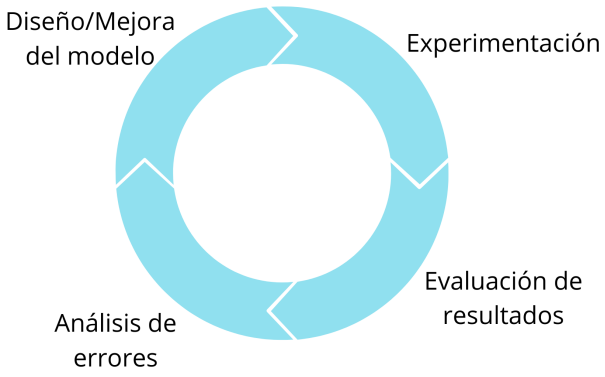

Figura 4: Ciclo de Desarrollo del modelo. Adaptado de: Deeplearning.ai

6 UN ENTORNO DE ENTRENAMIENTO PARA EL PARSING SEMÁNTICO

6.1. Entrenamiento del parser semántico

En esta sección, se proporcionará una visión general completa tanto del entorno como del agente de aprendizaje profundo por refuerzo. En este caso, el objetivo principal del agente es transformar una oración simple en una estructura de representación de discursos (DRS) correspondiente.

6.1.1. Entorno para la creación de la DRS

El entorno se encarga de establecer las condiciones necesarias para que el agente analice una oración y la transforme en una DRS. Cada episodio del entrenamiento dentro del entorno gira en torno a una única oración, llamada "oración focal". Al inicio, el entorno es inicializado con una DRS vacía. Las acciones disponibles implican el consumo de las palabras, donde una palabra se reemplaza por el símbolo de `mask` y, posteriormente, se incorpora a la DRS creando un referente y/o condiciones sobre esta. A continuación, presentamos los detalles de los componentes del entorno. En la Figura 5 se puede observar un ejemplo del procesamiento exitoso de una oración focal mostrando cómo las acciones modifican los estados en el entorno:

- **Lista de oraciones focales:** Se compiló una lista de 10 nombres propios y 10 verbos intransitivos en presente. Cada oración focal sigue la estructura de `nombre propio` + `verbo`. Se seleccionaron 100 oraciones por la simplicidad del problema y para prevenir un posible overfitting en el modelo. De este conjunto, 80 se asignaron para episodios de entrenamiento, mientras que las 20 restantes se designaron para una prueba de generalización.
- **Estados crudos:** Una tupla que consiste en: (a) la lista de palabras (o símbolo `mask`) que componen la oración focal; (b) el índice de la palabra dentro de la lista elegible para ser consumida por la acción; y (c) la DRS que abarca la representación de las palabras que han sido procesadas hasta el momento.
- **Estados:** Un vector de 512 dimensiones derivado de la representación lineal del estado crudo. Esta se genera aplicando el autocodificador del transformer "distiluse-base-multilingual-cased-v1" de la librería Sentence Transformers a la entrada, que

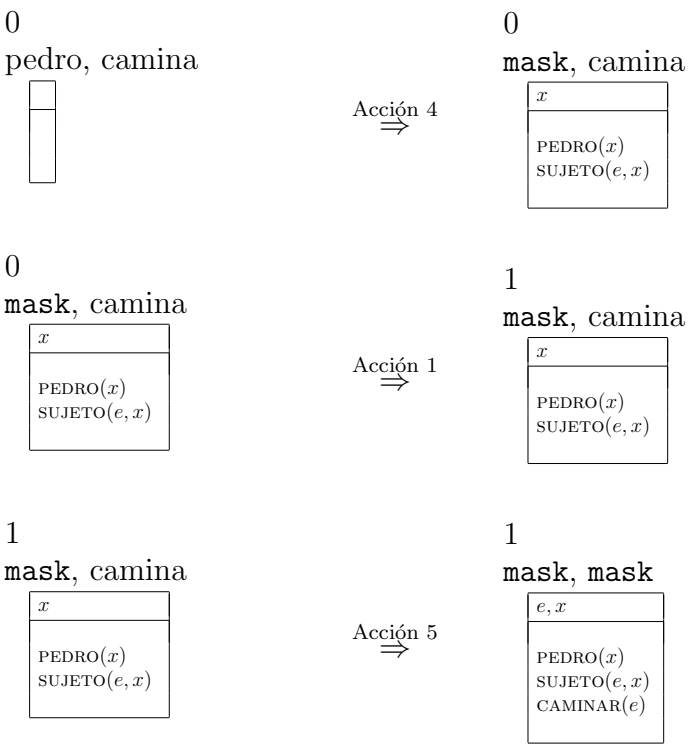

Figura 5: Ejemplo de un procesamiento exitoso de "Pedro camina". El estado inicial es una tupla con índice 0, las palabras de la oración y una DRS vacía. A continuación, el agente realiza la acción 4, procesando la palabra "Pedro" e incluyendo un referente x y las condiciones PEDRO(x) y SUJETO(e, x) en la DRS. La fila central muestra el efecto de la acción 1, que cambia el índice de 0 a 1. Finalmente, la fila inferior muestra el efecto de la acción 5, en la que la palabra "camina" es procesada y se incluye el referente e y la condición CAMINAR(e) en la DRS. Fuente: Elaboración propia.

comprende los caracteres en la representación lineal de los elementos (a)–(c) de un estado crudo [32].

- **Acciones:** Hay cinco acciones disponibles:

  1. Mover el índice a la derecha.
  2. Mover el índice a la izquierda.
  3. Interpretar la palabra del índice como un sustantivo, añadiendo un referente de tipo objeto y un predicado de sustantivo a la DRS.
  4. Interpretar la palabra del índice como el sujeto de un verbo, añadiendo un referente de tipo objeto, un predicado de sustantivo y el predicado de sujeto al la DRS.
  5. Interpretar la palabra del índice como un verbo, añadiendo un referente de tipo evento y un predicado de verbo a la DRS.

- **Recompensas:** Al finalizar la construcción de la DRS, el entorno intenta responder a la pregunta "¿quién + `verbo`?" siguiendo el procedimiento previamente explicado en la sección 4.2.2. Inicialmente, el entorno identifica los posibles candidatos capaces de responder la pregunta planeada. En este contexto, un candidato es una respuesta a la pregunta si esta es una consecuencia lógica de la representación de la oración focal. Si la respuesta correcta está entre estos candidatos, el agente es recompensado con una puntuación de 100; de lo contrario, incurre en una penalización de -100.

  Así mismo, durante la fase de construcción de la DRS, cada acción tiene un costo de -1. Además, el agente puede enfrentar una penalización de -10 por cualquiera de la siguientes acciones:

  - Intentar un movimiento hacia la izquierda estando posicionado en el extremo izquierdo.
  - Intentar un movimiento hacia la derecha estando posicionado en el extremo derecho.
  - Intentar procesar una palabra que ya ha sido procesada y está marcada como `mask`.

6.1.2. Deep Q-Network

El objetivo de la red neuronal es que el agente decida una de las posibles acciones a realizar. Para esto, la entrada de esta es un tensor de 512×1 obtenido al codificar el estado en vectores (embeddings), como se definió en la sección 6.1.1. En la primera capa lineal de la red, los embeddings de 512 dimensiones se proyectan a una salida de 64 dimensiones, seguida de una función de activación sigmoidal que introduce no linealidad en la red. Posteriormente, la capa de salida consiste en otra capa lineal que toma la salida de la capa anterior (64 dimensiones) y la transforma en una salida final de 5 dimensiones, representando una única salida para cada acción válida que el agente puede tomar.

6.2. Configuración experimental

Para la configuración experimental, se inició cada episodio seleccionando aleatoriamente una de las 80 oraciones focales del conjunto de entrenamiento y se construyó el

entorno centrado en ella. Posteriormente, se le asignó al agente un máximo de 10 rondas para actuar en este entorno. Esto debido a que, como las oraciones focales siguen la estructura `nombre propio` + `verbo`, el procesamiento exitoso de la oración debería ser como se observa en la Figura 5, es decir, obtener la DRS en 3 pasos. El agente fue entrenado durante 100 episodios.

La Deep Q-Network usada en nuestros experimentos incorporó una memoria de repetición priorizada con una capacidad de 100 y un parámetro de priorización (ω) establecido en 2. En cada ronda, después de que la memoria del agente estuviera llena, se llevó a cabo un entrenamiento utilizando el optimizador de Adam durante 3 épocas, empleando minilotes (mini-batches) de tamaño 32 y una taza de aprendizaje (α) establecida en 0.1.

Para evaluar la estabilidad del proceso repetimos el ciclo de entrenamiento 30 veces.

7 RESULTADOS Y DISCUSIÓN

7.1. Barrido de hiper-parámetros

Se llevó a cabo un barrido de hiper-parámetros para evaluar el rendimiento de diversos agentes. Para ello, utilizamos $\gamma = \{0{,}8, 0{,}9, 0{,}99\}$ y $\epsilon = \{0{,}1, 0{,}05, 0{,}01\}$, y combinamos estos valores en cada una de las 30 simulaciones realizadas. Con estos resultados, obtuvimos la recompensa promedio, la desviación estándar de las recompensas y el porcentaje de terminación promedio, como se ilustra en la tabla 1.

γ	ϵ	Recompensa Promedio	Desviación Estándar	Terminación promedio
0.90	0.01	15.93	37.89	29.24
0.90	0.05	11.27	43.87	29.51
0.99	0.05	4.36	38.10	22.87
0.80	0.05	4.03	37.25	22.44
0.90	0.10	3.86	42.60	26.00
0.80	0.10	3.24	35.41	21.89
0.99	0.01	2.65	34.65	20.56
0.80	0.01	-2.15	29.64	16.73
0.99	0.10	-8.08	28.23	12.09

Tabla 1: Resultados promedios del barrido de hiper-parámetros

Podemos observar que hay una inestabilidad en los resultados y que el desempeño del agente no es el adecuado. Más adelante, en la sección 7.3 se hablará de ello.

Además, se calcularon las recompensas totales y el porcentaje de terminación del mejor modelo para cada conjunto de hiper-parámetros. En la Tabla 2, se observa que el porcentaje de terminación en 4 de las 8 combinaciones es del 100 %, junto con recompensas totales que reflejan el buen desempeño del agente.

γ	ϵ	Recompensa total	Terminación
0.80	0.10	98.00	100.0
0.90	0.10	98.00	100.0
0.90	0.01	98.00	100.0
0.99	0.05	98.00	100.0
0.80	0.05	95.94	98.0
0.99	0.01	91.50	95.0
0.90	0.05	89.04	92.0
0.80	0.01	81.32	88.0
0.99	0.10	81.32	88.0

Tabla 2: Resultados del mejor agente para cada combinación de hiper-parámetros

7.2. Desempeño del mejor agente

En la Figura 6 se ilustra la resolución de preguntas mediante SAT-solvers para la oración "Pedro camina". Este procedimiento sería similar para cualquier oración de la forma "nombre propio + verbo". Para obtener la respuesta a la pregunta "¿quién camina?", se implementó la clase Modelo, la cual extrae las entidades y predicados de la representación semántica de la oración (DRS) proporcionada por el agente.

En el ejemplo de la Figura 6, las entidades obtenidas son "Pedro" y "Ev_caminar", que representan dos tipos de entidades: individuo y evento, respectivamente. Al formular la pregunta "¿quién?", se buscan las entidades del tipo "individuo" para identificar los posibles candidatos. Si la pregunta fuera "¿qué hace Pedro?", se buscarían las entidades del tipo "evento". Posteriormente, se crea la fórmula lógica adecuada para responder a la pregunta inicial.

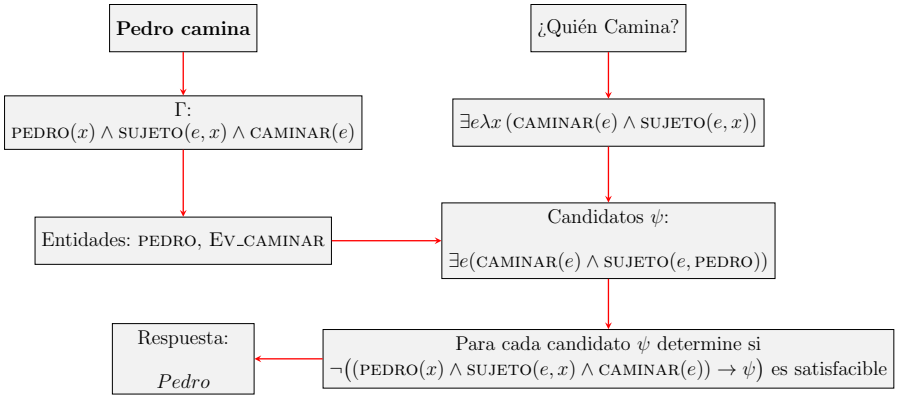

Figura 6: Resolución de preguntas mediante SAT-solvers para "Pedro camina". Fuente: Elaboración propia.

7.2.1. Generalización a nuevos casos

Para evaluar la capacidad del agente más allá de sus datos de entrenamiento, utilizamos las 20 oraciones reservadas en el conjunto de prueba. Realizamos 100 episodios, seleccionando aleatoriamente una oración focal del conjunto de prueba para cada episodio, con el agente equipado con la configuración de parámetros más efectiva encontrada durante el entrenamiento. La recompensa promedio a lo largo de estos episodios fue de 98 puntos, lo que indica un rendimiento impecable del agente en estos escenarios nuevos.

7.2.2. Generalización a nuevas preguntas

Hicimos ajustes menores al entorno para evaluar la DRS con una consulta diferente, utilizando las 20 oraciones del conjunto de prueba. Específicamente, empleamos la pregunta "¿qué hace `nombre propio`?" con los datos de prueba. Notablemente, incluso sin entrenamiento adicional, la DRS generada por el agente proporcionó consistentemente respuestas precisas a esta pregunta previamente no encontrada, logrando una tasa de éxito del 100 % como se observa en la tabla 2.

7.3. Entrenamiento y estabilidad

Con los resultados obtenidos del barrido de hiper-parámetros en la sección 7.1, se escogió $\gamma = 0{,}9$ y $\epsilon = 0{,}01$ para observar la inestabilidad que demuestra la tabla 1 y el mejor agente como se contempló en la tabla 2.

En la Figura 7, se observa el entrenamiento por 100 episodios del agente, donde se logra notar que en gran parte de estos, el agente obtiene la recompensa esperada el 73 % de las veces. Además, como lo observamos en la tabla 2, el agente logra una recompensa total de 98 y un porcentaje de terminación de 100 % en el conjunto de prueba.

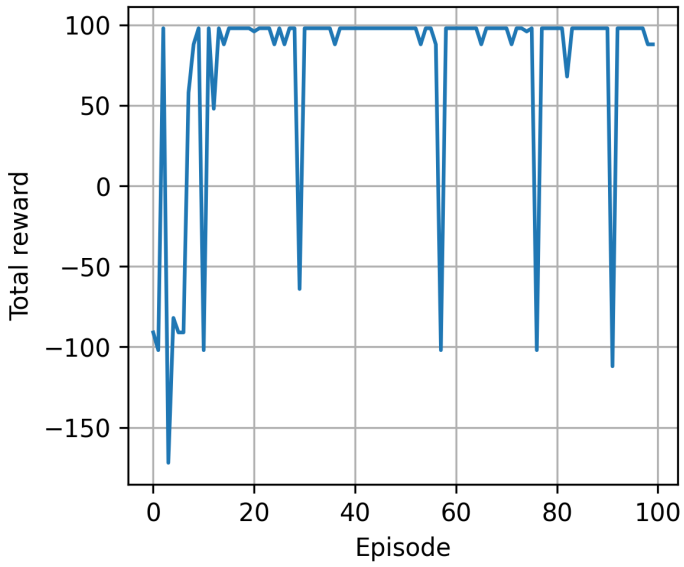

Figura 7: Resultados del entrenamiento de uno de los mejores agentes. El eje vertical representa la recompensa total obtenida por el agente durante un episodio. Esto fue para una simulación con los parámetros de $\gamma = 0{,}9$ y $\epsilon = 0{,}01$. Fuente: Elaboración propia.

En la Figura 8 podemos observar la recompensa total obtenida por el agente por

episodio. El eje vertical representa la recompensa total obtenida por el agente durante un episodio, promediada en 30 simulaciones. La característica más destacada que se desprende de las observaciones es la notable inestabilidad del proceso. Esta inestabilidad es discernible no sólo por los amplios márgenes de error sino también por la recompensa total media casi constante a lo largo de los episodios. Tal coherencia sugiere una dinámica en la que las simulaciones exitosas se ven contrarrestadas por casos en los que el desempeño del agente no es óptimo.

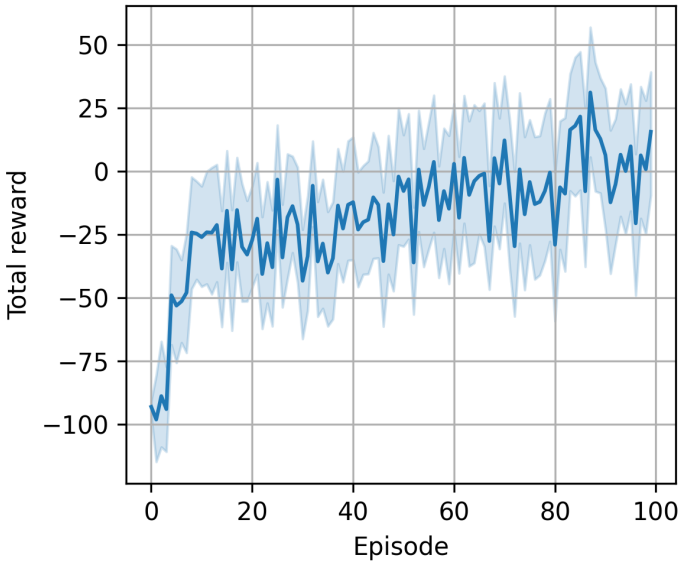

Figura 8: Resultados del entrenamiento de una Deep Q-Network en el entorno de análisis. El eje vertical representa la recompensa total obtenida por el agente durante un episodio, promediada en 30 simulaciones. Las áreas sombreadas alrededor de esta media representan un intervalo de confianza del 95 % determinado por la media y la desviación estándar. Fuente: Elaboración propia.

Debido a la inestabilidad mencionada anteriormente, decidimos profundizar sobre esto. Debido a que nuestras oraciones focales de entrenamiento son de la forma `nombre propio` + `verbo`, el procesamiento de cualquiera de estas debería ser el mismo. Así, para mayor brevedad, considere la oración "Pedro camina".

En la Figura 9 se puede observar los posibles caminos que puede tomar el agente. Se generan dos DRS distintas para una misma oración focal. Sin embargo, solo la del primer camino (cuando se realiza la acción 4) corresponde a la DRS correcta, que permite resolver la pregunta objetivo, como se explicó anteriormente. Para entender por qué el agente es capaz de llegar a los estados finales F y G, se decidió realizar un PCA de 2 dimensiones sobre los embeddings de los estados mencionados en la Figura 9.

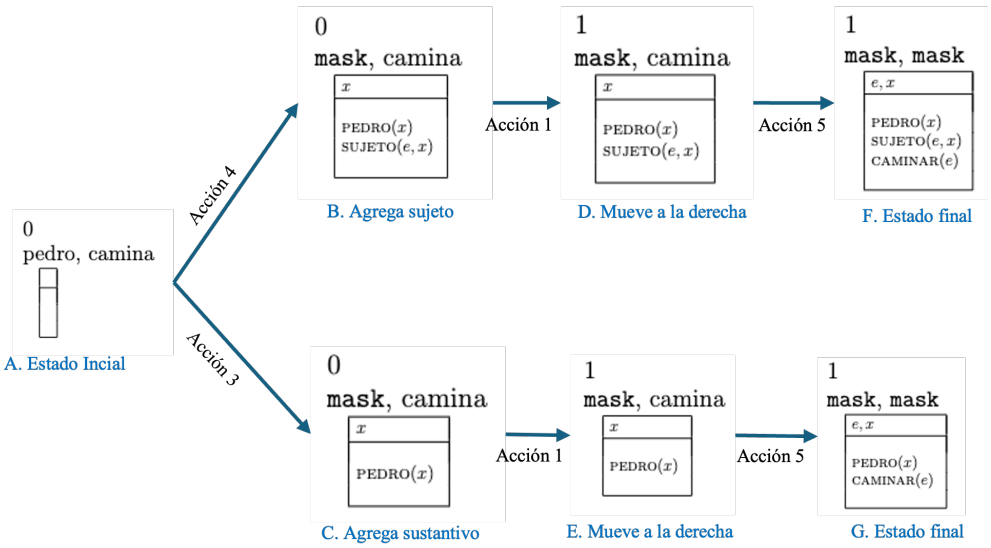

Figura 9: Ejemplo de los posibles caminos que puede tomar el agente al crear la DRS de una oración focal. El primer camino es cuando en el índice 0 procesa la palabra "Pedro" como sujeto y crea la DRS correspondiente; el segundo es cuando por el contrario es cuando procesa la misma palabra como un sustantivo. Fuente: Elaboración propia.

En la Figura 10, observamos que algunos estados se solapan, es decir, sus embeddings son tan similares que parecen casi idénticos. Un ejemplo notable es el solapamiento entre los estados D y G, que, a pesar de ser claramente distintos, presentan embeddings muy similares. El estado D corresponde a mover a la derecha, mientras que el estado G representa un estado final de una DRS incorrecta, indicando recompensas distintas para el agente. Esta similitud en los embeddings hace que el agente confunda estos estados, contribuyendo a la inestabilidad en el entrenamiento. De manera similar, los estados B y E, aunque no son terminales, se confunden debido a sus embeddings semejantes, lo que también añade inestabilidad al entrenamiento al representar acciones completamente distintas.

La similitud en los embeddings observada en la Figura 10, donde algunos estados se solapan y se confunden entre sí, sugiere que esta situación podría mejorarse mediante un ajuste fino (fine-tuning). Al aplicar fine-tuning, podríamos ajustar de manera más precisa los embeddings, diferenciando mejor entre estados como D y G, o B y E, que actualmente se confunden debido a su semejanza. Este ajuste permitiría al agente distinguir con mayor claridad entre estados que representan acciones y recompensas diferentes, lo que potencialmente reduciría la inestabilidad en el entrenamiento y mejoraría el desempeño del agente.

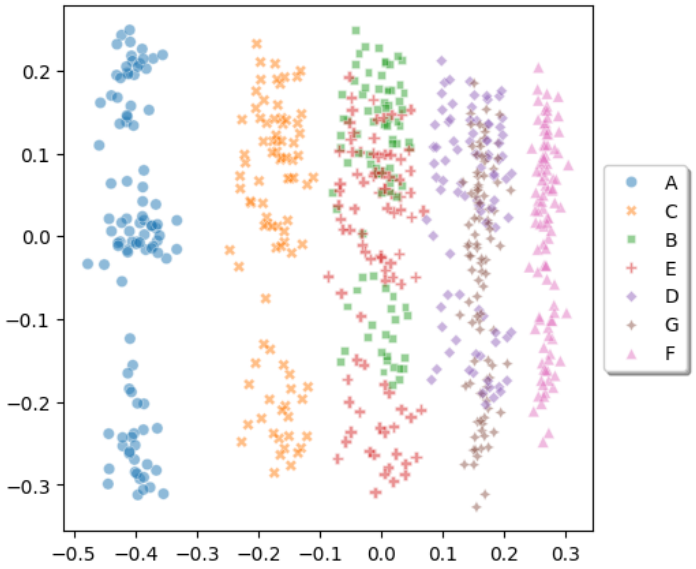

Figura 10: Resultados al visualizar mediante PCA los embeddings de los estados que se pueden generar al procesar una oración focal de dos maneras distintas. Utilizando PCA sobre los embeddings de las palabras de entrenamientos, logramos visualizar los estados que se pueden obtener procesando diferentes oraciones focales de la forma `nombre propio` + `verbo`. En las etiquetas se puede observar los estados nombrados en la Figura 9. Fuente: Elaboración propia.

Además, en este momento, dado que estamos tratando con un problema sencillo, no es de interés inmediato la capacidad de procesamiento que pueda llegar a tener el modelo. Sin embargo, al implementar el fine-tuning en un futuro, se analizaría la capacidad de procesamiento necesaria para entrenar el modelo a gran escala, asegurando así su eficiencia y eficacia en escenarios más complejos.

8 CONCLUSIONES

Sostenemos que la complejidad inherente a la conversión del lenguaje natural en representaciones lógicas presenta un impedimento significativo para su adopción generalizada debido a la necesidad de una amplia experiencia lingüística y lógica del ingeniero del conocimiento. Además, los avances recientes en el aprendizaje automático, en particular la arquitectura de transformers, han logrado evolucionar el campo al mostrar capacidades excepcionales de generalización. Sin embargo, estos modelos a menudo carecen de fiabilidad en tareas de inferencia lógica.

Nuestra metodología emplea el aprendizaje profundo por refuerzo para entrenar a un agente capaz de aprender autónomamente las reglas que rigen este proceso de conversión. Los resultados iniciales de nuestro estudio son prometedores, evidenciando la competencia del agente en la generación de Estructuras de Representación del Discurso adecuadas para abordar consultas básicas.

Nuestros esfuerzos futuros se centrarán en investigar diferentes arquitecturas de aprendizaje profundo por refuerzo y configuraciones de hiperparámetros para mejorar la estabilidad del aprendizaje. Así mismo, se adelantarán tareas de fine-tuning para que los estados se logren diferenciar más, facilitando así el entrenamiento del agente. Además, pretendemos ampliar el alcance del entorno para acomodar una gama más amplia de estructuras de oraciones, aumentando así la aplicabilidad y eficacia de nuestra metodología en escenarios del mundo real.

9 REFERENCIAS

[1] J. Eisenstein, *Introduction to Natural Language Processing*. MIT Press, 2019.

[2] D. Jurafsky and J. Martin, *Speech and Language Processing*. Prentice Hall, 2008.

[3] S. Ozdemir, *Quick Start Guide to Large Language Models: Strategies and Best Practices for Using ChatGPT and Other LLMs*. Addison-Wesley Professional, 2023.

[4] H. Kamp and U. Reyle, *From Discourse to Logic*. Dordrecht: Kluwer, 1993.

[5] F. van Harmelen, V. Lifschitz, and B. Porter, Eds., *Handbook of Knowledge Representation*. Amsterdam: Elsevier, 2008.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[7] A. Traylor, R. Feiman, and E. Pavlick, "AND does not mean OR: Using formal languages to study language models' representations," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 158–167.

[8] N. Kassner, B. Krojer, and H. Schütze, "Are pretrained language models symbolic reasoners over knowledge?" in *Proceedings of the 24th Conference on Computational Natural Language Learning*, R. Fernández and T. Linzen, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 552–564.

[9] V. Basmov, Y. Goldberg, and R. Tsarfaty, "Simple linguistic inferences of large language models (llms): Blind spots and blinds," 2024.

[10] M. Minsky, "A framework for representing knowledge," in *The Psychology of Computer Vision*, P. Winston, Ed. McGraw-Hill, 1975.

[11] D. McDermott, "A critique of pure reason," *Computational Intelligence*, vol. 3, pp. 151–160, 1987.

[12] L. T. F. Gamut, *Logic, Language and Meaning Vol. 2.* University of Chicago Press, 1991.

[13] E. T. Mueller, *Common Sense Reasoning.* Elsevier, 2006.

[14] J. Starc and D. Mladenić, "Joint learning of ontology and semantic parser from text," 2016. [Online]. Available: https://arxiv.org/abs/1601.00901

[15] X. Ho, A.-K. Duong Nguyen, S. Sugawara, and A. Aizawa, "Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps," in *Proceedings of the 28th International Conference on Computational Linguistics*, D. Scott, N. Bel, and C. Zong, Eds. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 6609–6625. [Online]. Available: https://aclanthology.org/2020.coling-main.580

[16] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," 2018. [Online]. Available: https://arxiv.org/abs/1809.09600

[17] M. Geva, D. Khashabi, E. Segal, T. Khot, D. Roth, and J. Berant, "Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 346–361, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:230799347

[18] D. Davidson, *Essays on Actions and Events: Philosophical Essays Volume 1.* Oxford, GB: Clarendon Press, 2001.

[19] M. van Lambalgen and F. Hamm, *The Proper Treatment of Events*, ser. Explorations in Semantics. Wiley, 2008.

[20] R. Montague, *Formal Philosophy: Selected Papers of Richard Montague.* New Haven: Yale University Press, 1974.

[21] B. Geurts, D. I. Beaver, and E. Maier, "Discourse Representation Theory," in *The Stanford Encyclopedia of Philosophy*, Spring 2020 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2020.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning*, 2nd ed. MIT Press, 2018.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[24] A. Zai and B. Brown, *Deep Reinforcement Learning in Action*. Manning Publications, 2020.

[25] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[27] V. Uc-Cetina, N. Navarro-Guerrero, A. Martin-Gonzalez, C. Weber, and S. Wermter, "Survey on reinforcement learning for language processing," *Artificial Intelligence Review*, vol. 56, no. 2, p. 1543–1575, Jun. 2022. [Online]. Available: http://dx.doi.org/10.1007/s10462-022-10205-5

[28] J. Bos, "Wide-coverage semantic analysis with Boxer," in *Semantics in Text Processing. STEP 2008 Conference Proceedings*. College Publications, 2008, pp. 277–286. [Online]. Available: https://aclanthology.org/W08-2222

[29] R. van Noord, L. Abzianidze, A. Toral, and J. Bos, "Exploring neural methods for parsing discourse representation structures," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 619–633, 2018. [Online]. Available: https://aclanthology.org/Q18-1043

[30] E. Andrade-Lotero, "Meaning and form in event calculus. msc. thesis," 2006.

[31] M. Binz and E. Schulz, "Using cognitive psychology to understand gpt-3," *Proceedings of the National Academy of Sciences*, vol. 120, no. 6, Feb. 2023. [Online]. Available: http://dx.doi.org/10.1073/pnas.2218523120

[32] Hugging Face, "sentence-transformers/distiluse-base-multilingual-cased-v1," 2024. [Online]. Available: https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v1

[33] L. Abzianidze, J. Bjerva, K. Evang, H. Haagsma, R. van Noord, P. Ludmann, D.-D. Nguyen, and J. Bos, "The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 242–247. [Online]. Available: https://aclanthology.org/E17-2039