

**FACULTAD DE INGENIERÍA**

**DEPARTAMENTO DE CIENCIA DE DATOS E INFORMÁTICA**

**CARRERA DE INGENIERÍA EN INFORMÁTICA**

**FORJANDO UNA REBELIÓN:**

**Construcción de un escenario de videojuego con sistema de diálogo IA en la Baja Edad Media Inglesa**

Proyecto de Título para optar el Título de Ingeniería en

Informática

**Esteban Adolfo Ponce Lira**

Profesor Guía: Dr. Mario Andrés Bruno Meléndez

Valparaíso, Chile

2024

**TABLA DE CONTENIDO**

[Resumen 4](#_Toc181566795)

[Abstract 4](#_Toc181566796)

[Introducción 5](#_Toc181566797)

[Capítulo 1: Descripción General 6](#_Toc181566798)

[1.1 Descripción del problema 6](#_Toc181566799)

[1.2 Objetivos del estudio 6](#_Toc181566800)

[Capítulo 2: Hablar sobre NLP, LLM y videojuegos en la educación? 7](#_Toc181566801)

[Capítulo 3: Metodología 8](#_Toc181566802)

[3.1 Configuración del chatbot en Rasa Open Source 8](#_Toc181566803)

[Capítulo 4: Desarrollo/Proceso? 16](#_Toc181566804)

[4.1 Instalación de Rasa Open Source 16](#_Toc181566805)

[4.2 Creación de un chatbot en Rasa Open Source 16](#_Toc181566806)

[4.3 Entrenamiento del chatbot en Rasa Open Source 16](#_Toc181566807)

[4.4 Construcción de un escenario en Unity 16](#_Toc181566808)

[4.5 Conexión del chatbot con el escenario de Unity 16](#_Toc181566809)

[4.6 Pruebas en el chatbot en Rasa Open Source 16](#_Toc181566810)

[Capítulo 5: Resultados 17](#_Toc181566811)

[Capítulo 6: Conclusiones 18](#_Toc181566812)

[Referencias 19](#_Toc181566813)

**ÍNDICE DE ILUSTRACIONES Y CUADROS**

# Resumen

# Abstract

# Introducción

# Capítulo 1: Descripción General

## Descripción del problema

## Objetivos del estudio

1. **Objetivo general**

Crear un escenario de un videojuego con una historia ambientada en la Baja Edad Media Inglesa.

1. **Objetivos específicos**

* Estudiar herramientas de NLP y LLM para implementar en un videojuego ambientado en la Baja Edad Media Inglesa.
* Construir una escena de videojuego medieval que incorpore un sistema de diálogo con IA, permitiendo que esta formule preguntas y evalúe las respuestas del jugador.
* Hacer pruebas del sistema de diálogo del videojuego construido con herramientas de NLP.

# Capítulo 2: ¿Hablar sobre NLP, LLM y videojuegos en la educación?

**Tokenizadores**

**Intenciones**

**Entidades**

**NLP**

**LLM**

**NLU**

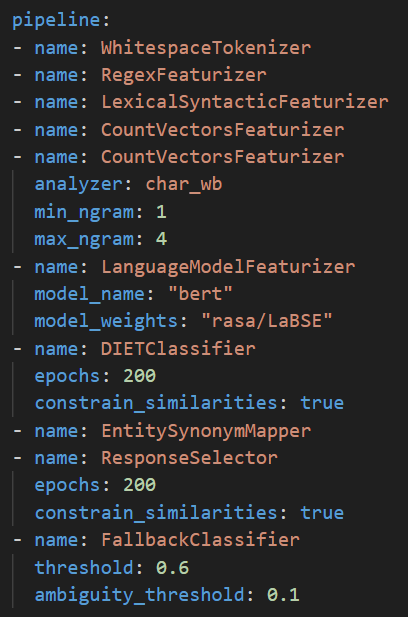
# Capítulo 3: Metodología

## Configuración del chatbot en Rasa Open Source

**Pipeline**

La pipeline de Rasa es una serie de componentes interconectados diseñados para procesar y entender las entradas de texto del usuario de manera efectiva. Esta estructura modular permite que el chatbot convierta el lenguaje natural en representaciones numéricas que pueden ser comprendidas por modelos de aprendizaje automático. A través de diversas técnicas de tokenización, featurización y modelado del lenguaje, la pipeline ayuda a extraer características relevantes de las interacciones, lo que permite clasificar intenciones y reconocer entidades. Cada componente de la pipeline tiene un propósito específico, desde la división del texto en tokens hasta la aplicación de modelos de lenguaje avanzados como BERT. Esta combinación de herramientas y enfoques asegura que el sistema sea capaz de manejar diálogos complejos y adaptarse a diferentes contextos de conversación. En resumen, una pipeline bien diseñada es crucial para el rendimiento general del chatbot, facilitando interacciones más precisas y satisfactorias para el usuario.

La pipeline que utilizaremos en nuestro proyecto será la pipeline predefinida de Rasa, con algunas modificaciones para adaptarla a nuestras necesidades específicas. Además, hemos incorporado el modelo preentrenado BERT, lo que potenciará la capacidad del sistema para comprender el contexto y las sutilezas del lenguaje natural.



WhitespaceTokenizer

* Descripción: Este tokenizador divide el texto de entrada en tokens (palabras) basándose en espacios en blanco.
* Uso: Es útil para mantener la estructura de las oraciones y se usa comúnmente en lenguajes donde los espacios en blanco son delimitadores naturales de palabras, como el inglés o el español.

RegexFeaturizer

* Descripción: Este componente utiliza expresiones regulares para extraer características del texto de entrada.
* Uso: Puede identificar patrones específicos en el texto, como números, fechas o correos electrónicos. Esto es útil para reconocer información estructurada.

LexicalSyntacticFeaturizer

* Descripción: Este featurizador extrae características léxicas y sintácticas de las oraciones.
* Uso: Se centra en características como la longitud de las palabras, el uso de mayúsculas, y otros elementos que pueden ser relevantes para la clasificación de intenciones y entidades.

CountVectorsFeaturizer

* Descripción: Este componente convierte el texto en vectores de recuento, que representan la frecuencia de aparición de palabras en el texto.
* Uso: Los vectores de recuento son útiles para modelos de aprendizaje automático, ya que convierten el texto en una representación numérica que puede ser procesada.

CountVectorsFeaturizer (analyzer: char\_wb)

* Descripción: Similar al anterior, pero este utiliza caracteres como unidades de análisis. Se generan n-gramas de caracteres en lugar de palabras, lo que permite captar patrones a nivel de carácter.
* Parámetros:
  + min\_ngram: 1 (se incluyen n-gramas de longitud mínima 1)
  + max\_ngram: 4 (se incluyen n-gramas de longitud máxima 4)
* Uso: Es útil para detectar errores tipográficos y patrones de escritura que no se pueden captar a nivel de palabra.

LanguageModelFeaturizer

* Descripción: Este componente utiliza un modelo de lenguaje preentrenado (en este caso, BERT) para extraer características del texto.
* Parámetros:
* model\_name: "bert" (indica que se utiliza BERT como modelo de lenguaje)
* model\_weights: "rasa/LaBSE" (específica el modelo de BERT que se usará, LaBSE es un modelo de embeddings de oraciones multilingües).
* Uso: Permite comprender mejor el contexto y las relaciones semánticas entre palabras en las oraciones.

DIETClassifier

* Descripción: Este es el clasificador principal para detectar intenciones y entidades en el texto.
* Parámetros:
* epochs: 200 (número de épocas para entrenar el modelo)
* constrain\_similarities: true (esto asegura que las similitudes entre las intenciones sean limitadas, mejorando la precisión del modelo).
* Uso: Permite a Rasa clasificar las intenciones del usuario y reconocer entidades en la conversación.

EntitySynonymMapper

* Descripción: Este componente mapea sinónimos de entidades.
* Uso: Ayuda a manejar diferentes formas de referirse a la misma entidad, mejorando la coherencia en la interpretación de las entradas del usuario.

ResponseSelector

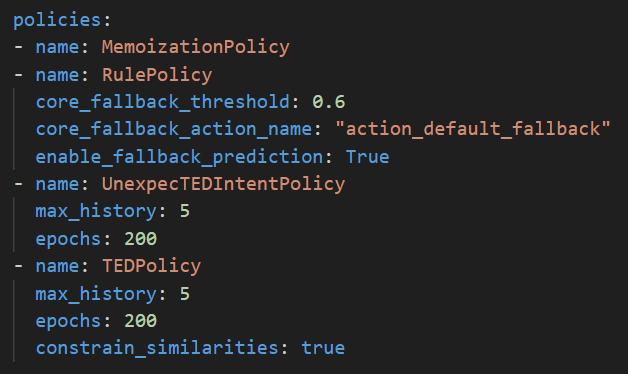
* Descripción: Este componente selecciona la respuesta adecuada a partir de un conjunto de posibles respuestas.
* Parámetros:
* epochs: 200 (número de épocas para entrenar el modelo)
* constrain\_similarities: true (asegura que las similitudes entre respuestas se mantengan dentro de límites, mejorando la precisión en la selección de respuestas).
* Uso: Es útil para proporcionar respuestas contextualmente relevantes basadas en la entrada del usuario.

FallbackClassifier

* Descripción: Este clasificador se utiliza para manejar entradas que no se pueden clasificar con alta confianza.
* Parámetros:
* threshold: 0.6 (si la confianza de la predicción es inferior a este umbral, se considera un fallback)
* ambiguity\_threshold: 0.1 (se usa para detectar ambigüedad en las predicciones).
* Uso: Permite redirigir el flujo de conversación cuando el modelo no está seguro, lo que puede ser útil para gestionar errores o consultas fuera de contexto.

**Policies**

Las políticas de Rasa son componentes fundamentales en el diseño de diálogos, ya que determinan cómo el modelo responde a las entradas del usuario y cómo gestiona el flujo de la conversación. A través de una combinación de enfoques basados en reglas y aprendizaje automático, estas políticas permiten al chatbot adaptarse a diversas situaciones, desde diálogos predecibles hasta interacciones más inesperadas. Cada política desempeña un papel específico, ya sea recordando patrones de conversación pasados, aplicando reglas definidas, o manejando intenciones no anticipadas. Esto resulta en un sistema de diálogo que no solo es capaz de comprender y procesar la información del usuario de manera efectiva, sino que también responde de forma coherente y relevante. En última instancia, la implementación adecuada de estas políticas contribuye a crear un chatbot más inteligente y robusto, mejorando la experiencia del usuario y facilitando interacciones más fluidas.



MemoizationPolicy

* Descripción: Esta política recuerda las historias de conversación pasadas y utiliza esta información para predecir las acciones futuras basadas en situaciones similares que ha encontrado anteriormente.
* Uso: Es eficaz en escenarios donde el flujo de la conversación es predecible y se repiten patrones. Puede ayudar a manejar interacciones comunes de manera más eficiente al reutilizar las respuestas y acciones previamente aprendidas.

RulePolicy

* Descripción: Esta política se utiliza para manejar reglas definidas explícitamente que guían el flujo de la conversación.
* Parámetros:
* core\_fallback\_threshold: 0.6 (si la confianza de la predicción es inferior a este umbral, se activa un comportamiento de fallback).
* core\_fallback\_action\_name: "action\_default\_fallback" (nombre de la acción que se ejecutará cuando se active el fallback).
* enable\_fallback\_prediction: True (habilita la predicción de fallback, lo que significa que la política puede prever situaciones en las que se necesita un comportamiento de fallback).
* Uso: Se utiliza para garantizar que se sigan ciertos patrones en la conversación y para manejar casos donde se requiere una respuesta específica en situaciones definidas. Esto es útil para asegurar un control sobre diálogos críticos y mejorar la experiencia del usuario.

UnexpecTEDIntentPolicy

* Descripción: Esta política está diseñada para manejar intenciones inesperadas, es decir, aquellas que no fueron anticipadas en el entrenamiento del modelo.
* Parámetros:
* max\_history: 5 (define cuántas interacciones anteriores se consideran al hacer predicciones).
* epochs: 200 (número de épocas para entrenar el modelo).
* Uso: Es útil en situaciones en las que los usuarios pueden desviarse del flujo de conversación esperado, permitiendo al modelo adaptarse a estas entradas no anticipadas y manejar diálogos más dinámicos y flexibles.

TEDPolicy

* Descripción: Esta política se basa en el modelo de Transformador de Diálogo (TED) y está diseñada para aprender de las historias de conversación.
* Parámetros:
* max\_history:5 (define cuántas interacciones anteriores se consideran al hacer predicciones).
* epochs: 200 (número de épocas para entrenar el modelo).
* constrain\_similarities: true (esto asegura que las similitudes entre acciones se mantengan dentro de límites específicos, lo que puede ayudar a mejorar la precisión del modelo).
* Uso: Se utiliza para predecir la siguiente acción en función de las interacciones anteriores y es útil en diálogos más complejos, donde las decisiones deben basarse en el contexto acumulado de la conversación.

**Corpus**

El corpus en Rasa se refiere al conjunto integral de datos de entrenamiento que permite al modelo comprender y gestionar interacciones en lenguaje natural. Este corpus está compuesto principalmente por tres componentes fundamentales: NLU, Historias y Reglas, cada uno desempeñando un papel crucial en el aprendizaje del sistema.

**NLU**

El archivo nlu.yml proporciona ejemplos de intenciones y entidades que el modelo debe aprender a reconocer, asegurando que el chatbot pueda interpretar correctamente las solicitudes de los usuarios. Este componente es esencial para la comprensión del lenguaje natural, ya que abarca la diversidad de formas en las que los usuarios pueden expresar sus necesidades.

El NLU utilizado en este proyecto está compuesto principalmente por 14 intenciones que representan posibles respuestas, 14 intenciones adicionales que se emplean como mensajes ocultos para activar las preguntas, y una intención destinada a manejar mensajes fuera de alcance.

**Historias**

El archivo stories.yml establece narrativas de conversación que ilustran cómo debería desarrollarse una interacción típica. Al modelar secuencias de diálogo, este componente ayuda al modelo a predecir acciones y a reaccionar de manera coherente en función del contexto y las intenciones del usuario.

Las historias en este proyecto abarcan 14 flujos de diálogo diferentes, cada uno diseñado para gestionar las 14 preguntas que el chatbot planteará a los usuarios. El formato de cada story comienza con un script que envía un mensaje oculto, que provocara que el chatbot realice una pregunta, el usuario responde la pregunta y finalmente se evalúa si la respuesta del usuario es correcta o incorrecta.

**Reglas**

El archivo rules.yml complementa las historias al definir reglas explícitas para el comportamiento del chatbot en situaciones específicas. Estas reglas aseguran que el sistema mantenga un flujo de conversación controlado y predecible, lo que es fundamental para gestionar interacciones críticas o recurrentes.

En este proyecto, se utilizarán las reglas para gestionar la detección de mensajes no clasificados y para implementar comportamientos de fallback. Estas reglas permitirán al chatbot reaccionar de manera adecuada cuando se encuentre con entradas que no se ajustan a las intenciones previamente definidas, asegurando que el sistema pueda manejar situaciones inesperadas de forma efectiva, si la entrada no se ajusta a las intenciones definidas en el NLU, el chatbot responderá con el mensaje “Incorrecto”.

# Capítulo 4: Desarrollo/Proceso?

## Instalación de Rasa Open Source

## Creación de un chatbot en Rasa Open Source

## Entrenamiento del chatbot en Rasa Open Source

## Construcción de un escenario en Unity

## Conexión del chatbot con el escenario de Unity

## Pruebas en el chatbot en Rasa Open Source

# Capítulo 5: Resultados

# Capítulo 6: Conclusiones

# Referencias

Formato de cita: [Número de cita] Autor, Título, Lugar de publicación, Editorial, Año de Publicación, Número de páginas.

Método: Usar formato IEEE y luego modificar las referencias para adaptarse al formato.