Proyecto Bootcamp Inteligencia Artificial

Felipe González Urrego

Desarrollo de la solución

1. Acceso a datos, identificación de variable objetivos y definición de hipótesis de partida.

1.1. Acceso a datos

Los datos son los brindados en el proyecto (link), en un primer acercamiento se descargan y se exploran superficialmente para obtener información de su estructura y contenido. Puesto que los datos no se encuentran en un formato habitual, primero es necesario llevarlos a un formato mas manejable como un dataframe, donde cada fila es una respuesta a una pregunta o subpregunta.

1.2. Identificación de variable objetivo

Clasificación de la pregunta por tipo: Si queremos clasificar las preguntas según su tipo (por ejemplo, "information", "cause", etc.), la variable objetivo sería la columna Type.

Predicción de la mejor respuesta: Si queremos predecir cuál es la mejor respuesta para una pregunta dada, necesitaríamos una variable que indique cuál es la mejor respuesta. Ya que esto no está explícitamente disponible en los datos, podríamos necesitar algún tipo de anotación adicional o suponer que todas las respuestas son igualmente correctas.

1.3. Definición de hipótesis

Hipótesis 1: Las preguntas médicas pueden clasificarse en diferentes categorías basadas en sus temas y enfoques (e.g., tratamiento, causa, diagnóstico, etc.).

Hipótesis 2: Utilizando técnicas de procesamiento de lenguaje natural (NLP) y aprendizaje automático, es posible entrenar un modelo que clasifique eficazmente las preguntas médicas y sugiera respuestas precisas.

2. Diseño del flujo de trabajo de las diferentes fases/etapas.

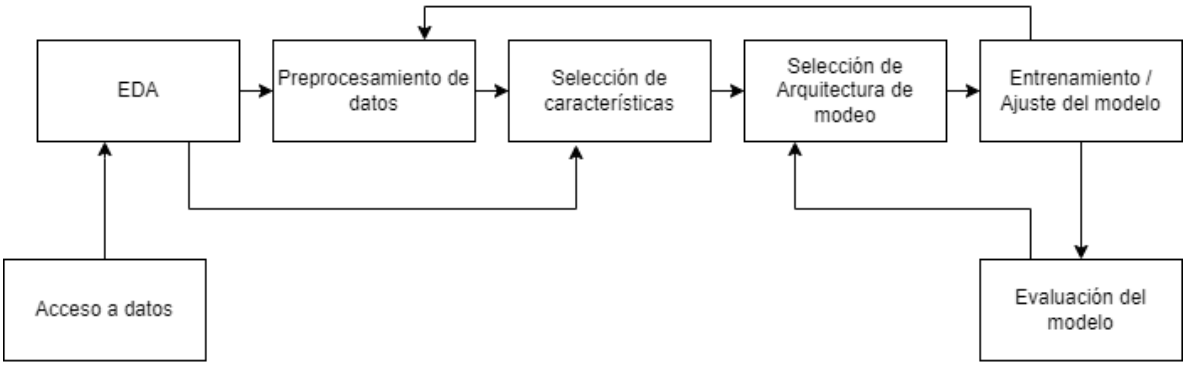

Ilustración 1. Diagrama de flujo de trabajo

2.1.1. Acceso y carga de datos

Objetivo: Cargar los datos desde los archivos XML y procesarlos para obtener un DataFrame estructurado.

- Leer los archivos XML desde el directorio de datos.
- Parsear los archivos XML para extraer las preguntas, temas, enfoques y respuestas.
- Almacenar los datos en un DataFrame de pandas.

2.1.2. Análisis exploratorio de datos (EDA)

Objetivo: Explorar la distribución de las preguntas, subpreguntas y respuestas para identificar patrones y características importantes.

- Realizar un análisis estadístico descriptivo de los datos.
- Visualizar la distribución de la longitud de las preguntas y respuestas.
- Identificar la frecuencia de temas y enfoques de las subpreguntas.

2.1.3. Preprocesamiento de datos

Objetivo: Limpiar y preparar los datos para el entrenamiento del modelo.

- Eliminar caracteres especiales y HTML del texto.
- Tokenizar y lematizar el texto.
- Convertir el texto en vectores utilizando técnicas como TF-IDF o embeddings.

2.1.4. Selección de características

Objetivo: Decidir qué características utilizar para el modelo.

- Evaluar diferentes características como el texto de las preguntas y respuestas, la longitud del texto, etc.
- Seleccionar las características más relevantes para la predicción.

2.1.5. Selección de modelo

Objetivo: Seleccionar el modelo de Deep Learning adecuado y decidir si hacer fine-tuning o entrenamiento desde cero.

- Evaluar diferentes modelos como BERT, TF-IDF + SVM, etc.
- Decidir si es necesario hacer fine-tuning de un modelo preentrenado o entrenar un modelo desde cero.

2.1.6. Entrenamiento del modelo

Objetivo: Entrenar el modelo seleccionado con los datos preprocesados y ajustar los hiperparámetros según sea necesario.

- Entrenar el modelo con el conjunto de entrenamiento.
- Ajustar los hiperparámetros.
- Validar el modelo con el conjunto de prueba.

2.1.7. Evaluación del modelo

Objetivo: Evaluar el rendimiento del modelo utilizando métricas técnicas y de negocio.

- Evaluar el modelo utilizando métricas seleccionadas.
- Comparar el rendimiento del modelo con las expectativas de negocio.

2.1.8. Implementación y despliegue

Objetivo: Implementar el modelo en un entorno de producción y crear endpoints para su uso en aplicaciones reales.

- Implementar el modelo en un servidor de producción.
- Crear endpoints de API para permitir que las aplicaciones accedan al modelo.
- Monitorear el rendimiento del modelo en producción y hacer ajustes si es necesario.

3. Análisis de datos y conclusiones preliminares.

Para el EDA se creó un notebook donde se detalla y concluyen las observaciones vistas en la exploración. En este notebook se exploró lo siguiente:

3.1.1. Distribución de la cantidad de respuestas por pregunta

Esto proporciona una idea de cuántas respuestas en promedio tiene cada pregunta, lo que es útil para entender la cantidad de información disponible por pregunta.

3.1.2. Distribución de la longitud de las respuestas y preguntas

Una longitud mayor puede indicar respuestas/preguntas más detalladas y completas.

3.1.3. Nube de palabras de las preguntas y respuestas

La nube de palabras ayuda a identificar los temas más frecuentemente discutidos en el dataset.

3.1.4. Relación entre la longitud de la pregunta y la longitud de la respuesta

Una correlación, aunque débil, sugiere que las preguntas más detalladas pueden llevar a respuestas más detalladas.

3.1.5. Distribución de las categorías de las preguntas

Identificar las categorías más frecuentes ayuda a focalizar los esfuerzos de análisis y modelado en los temas más comunes.

Tanto los gráficos como las conclusiones son visibles en este notebook.

4. Selección de métricas técnicas y de negocio.

4.1. Métricas técnicas

4.1.1. F1 score

F1 score es particularmente útil cuando hay un desequilibrio entre las clases y es importante considerar tanto los falsos positivos como los falsos negativos. En el dominio de la salud, donde tanto la precisión de la información como la cobertura completa de las respuestas relevantes son cruciales, el F1 score proporciona una métrica equilibrada para evaluar el rendimiento del modelo.

4.1.2. Recall

Proporción de instancias relevantes que han sido recuperadas sobre el total de instancias relevantes (verdaderos positivos + falsos negativos). importante cuando es crítico capturar la mayor cantidad de instancias relevantes posible, incluso si se incluyen algunos falsos positivos. En el contexto médico, un alto recall garantiza que la mayoría de las respuestas relevantes sean recuperadas, lo cual es vital para una cobertura completa de la información solicitada.

4.1.3. Precision

Proporción de instancias relevantes recuperadas (verdaderos positivos) entre el total de instancias recuperadas (verdaderos positivos + falsos positivos). crucial cuando las consecuencias de una predicción incorrecta (falso positivo) son altas. En el contexto de la salud, es importante que las respuestas proporcionadas sean correctas y relevantes para la pregunta original para evitar proporcionar información incorrecta al usuario.

4.2. Métricas de Negocio

4.2.1. Satisfacción del Usuario

Mide el nivel de satisfacción de los usuarios con las respuestas proporcionadas por el sistema. Se puede medir por medio de encuestas de satisfacción post-interacción, puntuaciones de satisfacción (por ejemplo, NPS - Net Promoter Score).

4.2.2. Tiempo de Respuesta

El tiempo promedio que toma el sistema para proporcionar una respuesta a una consulta. La obtenemos midiendo el tiempo desde que se recibe una consulta hasta que se proporciona una respuesta.

4.2.3. Retención de Usuarios

La proporción de usuarios que continúan utilizando el sistema después de su primera interacción. La proporción de usuarios que continúan utilizando el sistema después de su primera interacción.

5. Preprocesamiento de datos

En el prerprocesamiento limpiaremos y prepararemos nuestros datos para el modelado. Aquí aplicaremos técnicas para eliminar caracteres especiales, lematizar o hacer stemming, eliminar stopwords, convertir el texto en vectores y demás.

6. Selección de arquitectura de modelo

Para el diseño del enrutador de preguntas se definen 2 componentes basados en las variables objetivo. Un modelo Clasificador, el cuál será entrenado utilizando el modelo pre-entrenado "BiomedNLP-BiomedBERT-base-uncased-abstract-fulltext" este modelo será encargado de detectar el tipo de pregunta que hace el usuario y filtrar las respuestas posibles. El segundo componente será el uso de un modelo NER "ner-disease-ncbi-bionlp-bc5cdr-pubmed" ya entrenado en con datos médicos, este modelo se encargará de usar la variable "focus" para refinar aún mas el filtro que lleva a la respuesta de acuerdo a la entidad (enfermedad) que se hace consulta.

7. Evaluación del modelo

8. Diseño del flujo de trabajo fase de inferencia/producción

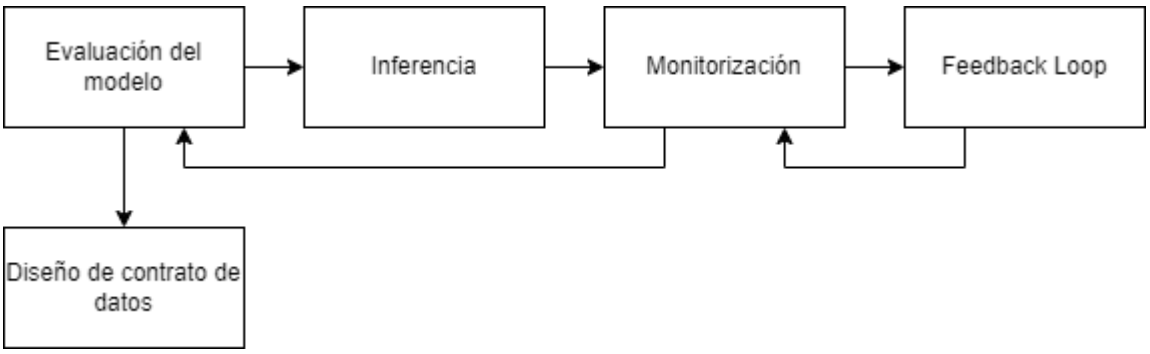

Ilustración 2 Diagrama flujo fase de inferencia

8.1.1. Inferencia

Implementar funciones para realizar predicciones en tiempo real.

8.1.2. Monitorización

Implementar métricas de rendimiento y monitorización continua.

8.1.3. Feedback Loop e Interacción con el Usuario

Diseñar e implementar mecanismos para recolectar retroalimentación del usuario.

9. Diseño y definición de contratos de datos entre componentes

Para la definición de contratos de datos se propone:

9.1.1. Componente: data_preparation.py

- Entrada: Archivos XML de datos de entrenamiento y test.
- Salida: DataFrame de pandas con las siguientes columnas:
    - qid: Identificador único de la pregunta.
    - message: Texto de la pregunta.
    - subject: Sujeto de la pregunta.
    - focus: Enfoque de la pregunta.
    - type: categoría de pregunta.
    - answers: Lista de respuestas a la pregunta.

9.1.2. Componente: text_cleaning.py

- Entrada: Texto a ser limpiado.
- Salida: Texto limpio y lemmatizado.

9.1.3. Componente: tr_model_training.py

- Entrada: DataFrame preparado con las preguntas y sus atributos (desde data_preparation.py).
- Salida: Modelo clasificador entrenado y tokenizadores.

Modelo: Objeto de modelo de clasificación entrenado (HuggingFace).

9.1.4. Componente: fr_model.py

- Entrada: pregunta del usuario al modelo NER
- Salida: entidad detectada por el modelo NER.

9.1.5. Componente: inference.py

- Entrada: Pregunta del usuario.
- Salida: Predicciones de los modelos y respuestas posibles.
    - Categorías: Lista de categorías predichas.
    - Enfoques: Lista de enfoques predichos.
    - Lista de respuestas a la pregunta del usuario.

9.1.6. Componente evaluation.py

- Entrada: Datos de test (DataFrame) y modelos entrenados.
- Salida: Métricas de evaluación (precisión, recall, F1-score).

9.1.7. Componente monitoring.py

- Entrada: Predicciones del modelo y etiquetas reales.
- Salida: Métricas de rendimiento (precisión, recall, F1-score) y registros de monitorización.

9.1.8. Componente feedback_loop.py

- Entrada: Retroalimentación del usuario (pregunta, predicciones, etiquetas reales).
- Salida: Ajustes en el modelo o los datos de entrenamiento.

10. Implementación de funciones/componentes para la preparación de datos.

En el módulo para preparación de datos leemos los datos provenientes de los archivos .xml de entrenamiento, se hace el correspondiente parseo y se hace uso del módulo de limpieza de datos para limpiar los datos y así retornar un dataframe limpio al modelo de entrenamiento.

11. Implementación de funciones/componentes para la inferencia de los modelos.

El módulo de inferencia se encarga de entrenar el modelo clasificador y realizar una inferencia sobre una pregunta realizada por el usuario.

12. Implementación de funciones/componentes para monitorizar el funcionamiento del modelo a partir de las métricas seleccionadas.

Se implementó un módulo de monitoreo que registra logs para evaluar el desempeño del modelo y tomar decisiones a partir de las métricas obtenidas.

13. Diseño e implementación (si aplica) de feedback loop o capacidad de interacción con el usuario.

Por ultimo se diseñó un modulo para el feedback loop, sin embargo, no fue implementado en el resultado final, es una mejora a realizar en el futuro