Inteligencia artificial avanzada para la ciencia de datos II

TC3007C.501

Reto Evaluación

Integrantes

Sebastian Rodriguez Salinas (A00827463)

Alan Mondragón Rivas (A01734565)

Elmer Osiel Ávila Vargas (A00826359)

Carlos E. Lucio Domínguez (A00828524)

Diego Solis Higuera (A00827847)

25 de noviembre de 2022

1. Introducción

La presente documentación realiza un análisis sobre los modelos realizados a lo largo del proyecto, en este habla de la métrica seleccionada para elegir los mejores modelos, se realiza la comparativa y evaluación a partir de los resultados.

Para el desarrollo del proyecto se crearon 2 tipos de modelos:

- Reconocimiento de Entidades Nombradas.
- Clasificación de Intenciones.

Se realizaron pruebas de modelos existentes, los cuales son ofrecidos como servicios por compañías terceras (en este caso Neuraan), dichas pruebas y resultados se analizan igualmente durante el documento y con ello se realiza la selección del mejor modelo para la implementación final.

2. Métricas Seleccionadas

La métricas seleccionadas para la evaluación de los modelos fueron 2:

- Precisión / Acuraccy
- Pérdida / Loss

A partir de la precisión podemos observar que tan certero es el modelo ante la predicción de distintos datos (entrenamiento y prueba), con ello podemos saber si el modelo es confiable o tiene grandes áreas de mejora. Por otro lado, con la pérdida podemos darnos cuenta del error de las predicciones, es decir, que tan alejados estamos de los resultados esperados, métrica que igualmente funge como indicador de que tan bueno o malo es el modelo.

3. Comparación y Evaluación de los Modelos

3.1. Modelo de Reconocimiento de Entidades Especiales

3.1.1. Modelo Entity Ruler vs Modelo NER

El modelo Entity Ruler de spaCy te permite definir patrones que serán reconocibles al momento de poner a prueba el modelo con oraciones o cadenas de texto específicas. Este modelo no tiene

un entrenamiento por iteraciones si no que simplemente se le brindan las entidades a reconocer y actúa como un diccionario de patrones que tiene eficiencia de búsqueda, es por ello que tiene un accuracy del 100% en el reconocimiento de entidades.

Por otro lado, el modelo NER (Named Entity Recognizer) si lleva un entrenamiento por iteraciones donde se implementa un algoritmo de machine learning para mejorar el aprendizaje pero se requiere de una inmensa cantidad de oraciones donde se ejemplifique la presencia de las entidades a reconocer. El uso de este modelo requeriría más tiempo tanto para la creación de los datos de entrenamiento como para el entrenamiento en cuestión de recursos computacionales, además de que sería complicado tener un accuracy muy alto debido a la limitación de datos de entrenamiento.

Evaluación:

Se selecciona el modelo **Entity Ruler** debido a que muestra un 100% de precisión además de que se adecua de manera perfecta a las necesidades del proyecto.

3.1.2. Modelo Entity Ruler vs Modelo de Detección de Entidades Especiales (Neuraan)

La detección de entidades permite ver la existencia de ciertas palabras o cadenas de palabras dentro de un texto, para nuestro proyecto es esencial reconocer dichas entidades específicas para poder clasificar la intención del usuario así como obtener algunas variables a utilizar para la extracción de información de la base de datos de INEGI.

Modelo de Detección de Entidades Especiales (Neuraan):


```
    "recieved": {
        "text": "Quiero estudiar en la Universidad de Yucatán o en la Modelo",
        "entities": {
            "UADY_CANONICAL": ["Universidad Autónoma de Yucatán", "UADY"],
            "MODELO_CANONICAL": ["modelo","universidad modelo","la modelo"]
        },
        "threshold": 0.65,
        "knowledge_base_origin": "request"
    },
    "result": {
        "detected": [
            ["UADY_CANONICAL",0.6551724137931034,"Universidad Autónoma de
Yucatán","universidad_de_yucatan_o_en_la_"],
            ["MODELO_CANONICAL",0.7142857142857143,"la modelo","la_a_mode"]
        ],
        "indexes": [
            ["UADY_CANONICAL",22,51,1],
            ["MODELO_CANONICAL",19,26,2]
        ]
    }
```


Figura 3.1.2.1. Ejemplo del Modelo de Detección de Entidades Especiales (Neuraan).

Inicialmente se probó con el Modelo de Detección de Entidades Especiales de Neuraan, sin embargo, este presenta algunas limitaciones respecto al comportamiento deseado e indicado para el proyecto, el servicio no permite la detección de datos únicamente numéricos seguidos de una cadena específica cuyo caso es necesario para el proyecto, en la imagen 3*.1.2.1. Ejemplo del Modelo de Detección de Entidades Especiales (Neuraan)* se observa que el modelo permite el reconocimiento de cadenas (entities) específicas, debido a ello se buscó una alternativa respecto a este.

Modelo Entity Ruler:

Por su parte, el modelo Entity Ruler permite la generación de diccionarios para el reconocimiento de patrones tales como el necesario para el proyecto, en la *Figura 3.1.2.2. Diccionario de Patrones a Reconocimiento* se observa que los primeros 4 patrones indican el reconocimiento de un dato numérico seguido por las unidades de distancias posibles a detectar (caso imposible de realizar con el servicio de Neuraan) cuyo etiqueta/entidad corresponde a DIS (Distancia).

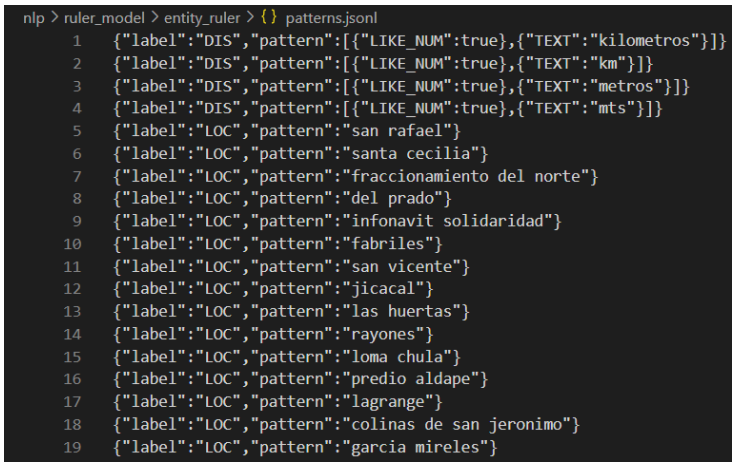

Figura 3.1.2.2. Diccionario de Patrones a Reconocimiento.

En la *Figura 3.1.2.3. Ejemplo Resultado del Modelo Entity Ruler* se observan los resultados del procesamiento de la oración "¿Cuáles son los oxxos a veintidós km de mi?", en la primer línea se aprecia el preprocesamiento de dicha oración que eliminó los acentos, signos de admiración y conversión de mayusculas a minusculas para hacer el análisis del lenguaje más sencillo, en las

siguientes líneas se aprecia el resultado, donde se muestra que se reconoció la entidad de EST (establecimiento) oxxo y la entidad DIS (distancia) 22 km.

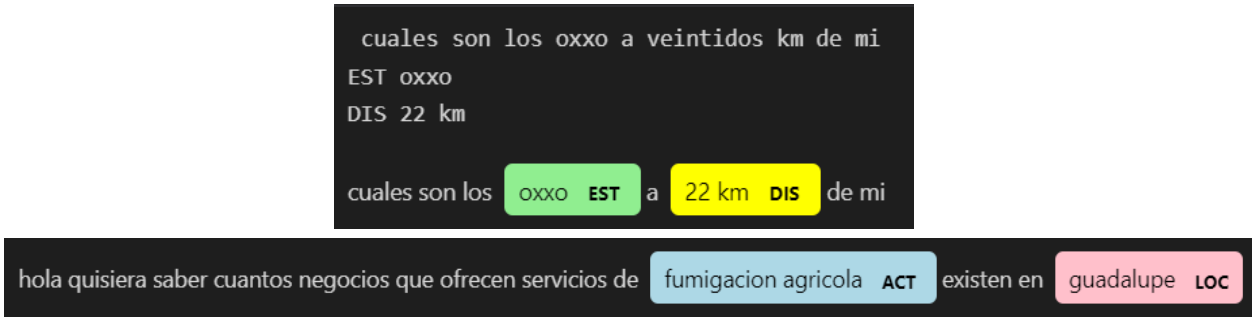

Figura 3.1.2.3. Ejemplos de salidas del Modelo Entity Ruler.

Evaluación:

Se seleccionó el modelo **Entity Ruler** debido a que tiene una precisión del 100% gracias a la posibilidad de crear un diccionario específico a partir de la base de datos con la que se cuenta. Por otro lado, se descartó la posibilidad de utilizar el Modelo de Detección de Entidades Especiales de Neuraan debido a las limitaciones que presentó.

3.2. Modelo de Clasificación de Intenciones

3.2.1. Modelo Text Categorizer (spaCy) vs Intent Classification de Neuraan

La clasificación de intenciones de nuestro chatbot se llevó a cabo mediante el modelo Text Categorizer de spaCy, el cual te permite clasificar cadenas de texto en categorías, siendo esencial en nuestro producto ya que nos permite identificar que tipo de búsqueda desea realizar el usuario. En un inicio se intentó realizar esto con el modelo de clasificación de intenciones de Neuraan, sin embargo las dificultades que se nos presentaron para diferenciar entre dos de las categorías/intenciones fue lo que nos llevó a realizar nuestro propio modelo.

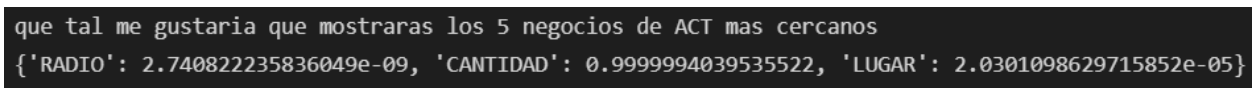

Figura 3.2.1.1. Ejemplo del modelo Text Categorizer para clasificación de la intención de tipo “cantidad”.

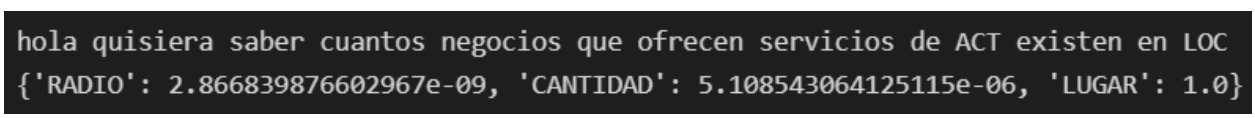

Figura 3.2.1.2. Ejemplo del modelo Text Categorizer para clasificación de la intención de tipo “radio”.

_base":{"CANTIDAD":["más cercana","más cercano","más cerca de mí","más cerca a mi ubicación","más cerca","más cercanos"],"RADIO":["radio de","kilómetros","km","metros de mi ubicación","mts de donde estoy"],"LUGAR":["en","dentro de"]},"input":"es","output":"es","knowledge_base_origin":"database"},"result":{"sentence":"Qué oxxos hay a 5 kilómetros de mi","equivalent":"Qué oxxos hay a 5 kilómetros de mi","categories":{"LUGAR":0.3357215523719787 6,"RADIO":0.2014373987913131 7,"CANTIDAD":0.0000176198136614402 76}}}

Figura 3.2.1.3. Ejemplo del modelo de Neuraan para clasificación de la intención de tipo "radio".

Durante el entrenamiento del modelo se guardó un historial de loss para analizar cuál fue el comportamiento del modelo respecto a la pérdida a lo largo de las iteraciones. El valor final de loss al guardar el modelo fue de *4.98*, el cual es relativamente bajo. En la figura 3.2.1.3 se puede apreciar una gráfica donde vemos que la pérdida fue disminuyendo conforme avanzaban las iteraciones pero después de la 4ta iteración ya no era tan relevante esta disminución.

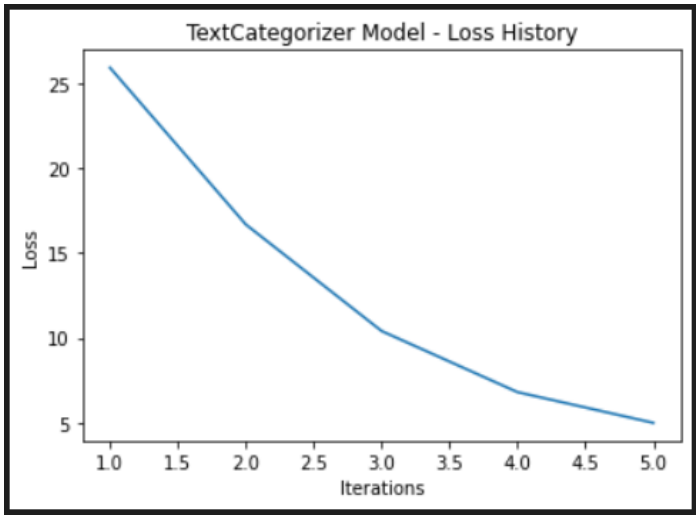

Figura 3.2.1.4. Historial de Loss durante las iteraciones para el entrenamiento del modelo Text Categorizer.

Así mismo, se utilizó el subset de datos para pruebas con la finalidad de medir la precisión del modelo. Como se puede ver en la figura 7.3.1.4, se consiguió un score de *100%* e incluso podemos ver el accuracy por categoría, donde también fueron valores de 100% de precisión.

```
textcat_model.accuracy_score(test_data)
```

Python

{'token_acc': 1.0, 'token_p': 1.0, 'token_r': 1.0, 'token_f': 1.0, 'ents_p': None, 'ents_r': None, 'ents_f': None, 'ents_per_type': None, 'cats_score': 1.0, 'cats_score_desc': 'macro AUC', 'cats_micro_p': 1.0, 'cats_micro_r': 1.0, 'cats_micro_f': 1.0, 'cats_macro_p': 1.0, 'cats_macro_r': 1.0, 'cats_macro_f': 1.0, 'cats_macro_auc': 1.0, 'cats_f_per_type': {'RADIO': {'p': 1.0, 'r': 1.0, 'f': 1.0}, 'CANTIDAD': {'p': 1.0, 'r': 1.0, 'f': 1.0}, 'LUGAR': {'p': 1.0, 'r': 1.0, 'f': 1.0}}, 'cats_auc_per_type': {'RADIO': 1.0, 'CANTIDAD': 1.0, 'LUGAR': 1.0}, 'speed': 42983.31753446295}

Figura 3.2.1.5. Accuracy test sobre los datos de prueba.

Evaluación:

Se seleccionó el modelo **Text Categorizer** debido a que presentó una mejor precisión en la clasificación de intenciones que el chatbot debería ser capaz de detectar.