



Inteligencia artificial avanzada para la ciencia de datos II

TC3007C.501

**Portafolio de Implementación:
Técnicas y Arquitecturas de Deep Learning**

Integrantes

Sebastian Rodriguez Salinas (A00827463)

Alan Mondragón Rivas (A01734565)

Elmer Osiel Ávila Vargas (A00826359)

Carlos E. Lucio Domínguez (A00828524)

Diego Solis Higuera (A00827847)

1 de diciembre de 2022

1. Introducción

La presente documentación realiza un análisis sobre los modelos de procesamiento de lenguaje natural (NLP) realizados a lo largo del proyecto. En este se habla profundamente de distintos aspectos, incluyendo la arquitectura de los modelos, la solución generada, ajustes en los modelos, análisis de resultados, métricas de evaluación, comparativas, entre otros. Con fines de contextualizar el documento para facilitar su comprensión, cabe mencionar que:

- El proyecto consiste en el desarrollo de un sistema de software para INEGI que le brinda al usuario la posibilidad de interactuar con un chatbot al que le puede solicitar que realice búsquedas de establecimientos y negocios en Nuevo León.
- El chatbot está motorizado por una API que sostiene 2 modelos de NLP (Reconocimiento de Entidades Nombradas y Clasificación de Intenciones) y algoritmos de búsqueda específicos sobre el dataset del INEGI. Las búsquedas resultantes se despliegan en un mapa dentro de la página web donde se encuentra el chatbot.

Se realizaron pruebas de modelos ya 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 del producto.

2. Arquitecturas utilizadas

2.1. Herramienta y lenguaje utilizado

Para llevar a cabo el desarrollo de ambos modelos implementados se utilizó el lenguaje de *python* con la librería de *spaCy*. De esta librería se utilizó específicamente el módulo de *EntityRuler* para el modelo de reconocimiento de entidades nombradas y *TextCategorizer* para el modelo de clasificación de intenciones.

2.2. Definición de la arquitectura

Respecto al modelo de clasificación de intenciones (TextCategorizer) se empleó la arquitectura base generada por la librería spaCy de la clase TextCatEnsemble; ésta emplea una red neuronal convolucional (CNN) con 2 capas de convolución, y un tamaño de entrada y salida de las capas de 64 de anchura. Es importante mencionar que además, la librería mencionada emplea el modelo “bolsa de palabras” para representar documentos e ignorar el orden de las palabras.

Por su parte, el modelo de reconocimiento de entidades nombradas no emplea una red neuronal para el entrenamiento; sino que a través de una arquitectura no entrenable se genera un documento basado en reglas que permite un reconocimiento eficiente de entidades dentro de una cadena de texto brindada.

2.3. Ajustes durante la implementación

En el apartado anterior se menciona que en el modelo TextCategorizer fue empleada la arquitectura base generada por la librería, esto debido a que comúnmente las librerías modernas proponen por defecto arquitecturas y valores en los parámetros de forma optimizada, donde muchas veces se obtienen buenos resultados; una vez que se probó dicha estructura el único ajuste que se propuso y se realizó fue la adición de *dropout* de 30% para evitar overfitting durante el entrenamiento.

2.4. Arquitecturas Probadas

Inicialmente se probó el servicio de clasificación de intenciones específicas de la compañía Neuraan; la arquitectura de este no se encuentra especificada en la documentación provista por ellos, se puede mencionar además que se consultó directamente a la empresa a través de correo y no se obtuvo respuesta respecto a la arquitectura. Sin embargo, es preciso mencionar que los resultados arrojados por dicho modelo fueron menos precisos que los resultados obtenidos por el clasificador de intenciones generado con spaCy, a raíz de ello se decidió implementar y refinar este último modelo.

3. Solución propuesta

3.1. Modelo de Reconocimiento de Entidades Nombradas

A través de la arquitectura mencionada para este modelo, se implementó una solución que permite identificar eficientemente patrones o entidades cuyo reconocimiento es fundamental para llevar a cabo el proceso de búsqueda de los establecimientos en el dataset de INEGI. El modelo es capaz de reconocer entidades de tipo *actividad*, *establecimiento*, *distancia*, *municipio* y *localidad*.

3.2. Modelo de Clasificación de Intenciones

Mediante la arquitectura de CNN mencionada, se creó un modelo que permite clasificar texto en categorías a partir de una cadena de caracteres proporcionada. Esta solución fue implementada ya que requeríamos un modelo que permitiera clasificar la intención de búsqueda del mensaje enviado por el usuario. Las categorías de intenciones que puede identificar el modelo son búsqueda por *radio*, *lugar* o *cantidad* de negocios.

4. Entradas de los modelos

4.1. Modelo de Reconocimiento de Entidades Nombradas

Respecto a este modelo se realizó una extracción de entidades nombradas del dataset del INEGI sobre establecimientos en Nuevo León. En primera instancia, se realizó un preprocesamiento sobre las columnas del dataset de donde se extraen las entidades, esto con la finalidad de hacer más eficaz las futuras búsquedas por intención sobre el dataset. Dicho preprocesamiento consistió en el uso de regex dentro de una función que convierte automáticamente el texto de las columnas a minúsculas, removiendo acentos, espacios extra y signos de puntuación.

```
def preprocess_text(sen):
    sentence = str(sen).lower()

    # -> NFD y eliminar diacríticos
    sentence = re.sub(
        r"([\u0300-\u036f]|n(?:!\u0303(?:![\u0300-\u036f])))[\u0300-\u036f]+", r"\1",
        normalize("NFD", sentence), 0, re.I
    )

    sentence = normalize('NFC', sentence) # -> NFC
    sentence = re.sub('[^\w]', ' ', sentence) # Remove punctuations
    sentence = re.sub(r'\s+', ' ', sentence) # Removing multiple spaces

    return sentence
```

Figura 4.1.1. Función para el preprocesamiento de las columnas de donde se extrajeron las entidades.

	nom_estab	raz_social	nombre_act	nomb_asent	municipio
0	agroacre san rafael	agroacre sa de cv	servicios de fumigacion agricola	san rafael	galeana
1	arquitectura y paisajismo	arquitectura y paisajismo sa de cv	servicios relacionados con el aprovechamiento ...	santa cecilia	monterrey
2	asociacion ganadera para nl	asociacion ganadera para nl	servicios relacionados con la cría y explotaci...	fraccionamiento del norte	paras
3	basculas tabesa	basulas tabesa sa de cv	servicios relacionados con la cría y explotaci...	del prado	monterrey
4	beef master	asociacion mexicana de criadores de ganado sa ...	servicios relacionados con la cría y explotaci...	infonavit solidaridad	guadalupe

Figura 4.1.2. Resultado del preprocesamiento de las columnas de donde se extrajeron las entidades.

Posteriormente, se crearon funciones para eliminar duplicados y limpiar substrings irrelevantes que podrían afectar el match en el reconocimiento de entidades. Por ejemplo, para los establecimientos se eliminaron cadenas de texto como SA de CV, S de RL de CV, entre otras.

```
def replaceEstRaz(s):
    arr = ["sa de cv", "sa de cv", "sa de cv", "sa de cv",
           "sa de cv", "sa de cv", "sa de cv", "sa de cv",
           "sa de cv", "sa de cv", "sa de cv", "sa de cv",
           "s de rl de cv", "s de rl de cv", "s de rl de cv", "s de rl de cv",
           "s de rl cv", "s de rl cv", "s de rl cv", "s de rl cv",
           "s de rl de cv", "s de rl de cv", "s de rl de cv", "s de rl de cv",
           "s de rl de cv", "s de rl de cv", "s de rl de cv", "s de rl de cv",
           "sc de rl de cv", "sc de rl de cv", "sc de rl de cv", "sc de rl de cv",
           "spr de rl de cv", "spr de rl", "s de pr de rl", "s pr de rl", "s de pr de rl de cv",
           "pi de cv", "pi de cv", "pi de cv", "pi de cv",
           "s fr de cv", "sc de c de rs de cv", "sp de rl", "q de cv", "s en nc de cv", "spp de rl", "sn de rl de cv",
           "sc de ap de rl de cv", "sc de ap de rl de cv", "sc de ap de rl de cv", "sc de ap de rl de cv", "sc de ap de rl de cv",
           "s de rl mi", "s de rl mi", "s de rl mi", "s de rl mi", "de rl mi", "ar de ic de rl", "aric rl sc",
           "s de rl de cv", "s de ap de rl de cv", "s pr de rl", "s pr de rl", "s de rl",
           "sc de rl", "sc de rl", "sc de rl", "sc de rl",
           "s de rl", "s de rl", "s de rl", "s de rl",
           "b de cv", "b de cv", "b de cv", "b de cv",
           "a de rl mf", "de rl de cv", "de rl de cv", "de rl", "de rl", "de rl", "de rl",
           "sa", "sa", "sa", "sa"]
    #
    for i in arr:
        s = s.replace(i, "")
    return s
```

Length With Duplicates: 186092
Length With Uniques: 9272
Length With Duplicates: 186092
Length With Uniques: 51
Length With Duplicates: 186092
Length With Uniques: 910
Length With Uniques After Applymap: 815
Length With Duplicates: 186092
Length With Uniques: 38037
Length With Uniques After Applymap: 37854
Length With Duplicates: 186092
Length With Uniques: 131848
Length With Uniques After Applymap: 131284
Merging est and raz:
Length before deleting duplicates: 169138
Length after deleting duplicates: 155157

Figura 4.1.3 y 4.1.4. Ejemplo de las funciones utilizadas para la limpieza de las entidades y demostración de la cantidad de datos duplicados que fueron eliminados.

Una vez conseguida la limpieza y preprocesamiento de las entidades que se utilizarían para el modelo de reconocimiento de entidades nombradas, se guardaron las entidades sobre archivos .txt que después serían leídos para añadir los patrones de reconocimiento al modelo.

```

nlp > training > est.txt
1 agroacren rafael
2 arquitectura y paisajismo
3 asociacion ganadera paras nl
4 basculas tabesa
5 beef master
6 biocombustibles sierra madre
7 biopec
8 captura de peces y otras especies
9 cortijo los alamares
10 descascaradora nueces procesadas de huertas los concabados
11 devar
12 empacadora de cebollas el rocio
13 frupack
14 generadora de apoyos comunes
15 granga acuicola protilapia
16 granja acuicola el civil
17 granja psicola general francisco villa
18 granja truticola el ciruelo
19 granja yumka
20 hacienda de bustamante
21 imporagri
22 inovacion alimenticia de mexico
23 la garnja
24 los cafugas
25 nueces y piñones del norte
26 oficina
27 oficina administrativa pronort
28 papas selectas rio fuerte
29 parques y vida silvestre de nuevo leon
30 pell radiant
31 pesca y captura de otros peces crustaceos moluscos y otras especies
32 piscicultura y otra acuicultura excepto camaronicultura

```

Figura 4.1.5. Archivo .txt donde se guardaron las entidades de tipo *establecimiento* extraídas del dataset.

4.2. Modelo de Clasificación de Intenciones

Referente a este modelo, se implementó una generación automática de datos para el entrenamiento. Dichos datos consisten en oraciones generadas por la combinación mediante código de diferentes palabras y formas en las que un usuario podría solicitar una búsqueda. Se creó un diccionario de palabras y un diccionario de reglas que después se utilizarían en una función para formar oraciones de las distintas combinaciones de palabras.

```

def getRand():
    return str(random.randint(1,500))

dictionary = {
    "pregind": ["que", "cual"],
    "pregindw": ["cual es", "donde esta", "muestrame", "muestra", "enseñame", "enseña", "ubicame", "ubica"],
    "pregpl": ["que", "cuales"],
    "pregplw": ["cuales son", "donde estan", "muestrame", "muestra", "enseñame", "enseña", "ubicame", "ubica"],
    "negpl": ["negocios que ofrecen servicios de ACT", "negocios de ACT", "tiendas de ACT", "lugares de ACT"],
    "negplw": ["los negocios que ofrecen servicios de ACT", "los negocios de ACT", "las tiendas de ACT", "los lugares de ACT"],
    "negplwque": ["los negocios que ofrecen servicios de ACT que", "los negocios de ACT que", "las tiendas de ACT que", "los lugares de ACT que"],
    "negcantind": ["EST", "negocio que ofrece servicios de ACT", "negocio de ACT", "tienda de ACT", "lugar de ACT"],
    "negcantindw": ["el EST", "la EST", "el negocio que ofrece servicios de ACT", "el negocio de ACT", "la tienda de ACT", "el lugar de ACT"],
    "negcantindwque": ["el EST que", "la EST que", "el negocio que ofrece servicios de ACT que", "el negocio de ACT que", "la tienda de ACT que", "el lugar de ACT que"],
    "negcantpl": [getRand()+" EST", getRand()+" negocios que ofrecen servicios de ACT", getRand()+" negocios de ACT", getRand()+" tiendas de ACT", getRand()+" lugares de ACT"],
    "negcantplw": ["los "+getRand()+" EST", "las "+getRand()+" EST", "los "+getRand()+" negocios que ofrecen servicios de ACT", "los "+getRand()+" negocios de ACT", "las "+getRand()+" tiendas de ACT", "los "+getRand()+" lugares de ACT"],
    "negcantplwque": ["los "+getRand()+" EST que", "las "+getRand()+" EST que", "los "+getRand()+" negocios que ofrecen servicios de ACT que", "los "+getRand()+" negocios de ACT que", "las "+getRand()+" tiendas de ACT que", "los "+getRand()+" lugares de ACT que"],
    "verbind": ["esta", "hay", "se encuentran", "se ubica", "existe"],
    "verbpl": ["estan", "hay", "se encuentran", "se ubican", "existen"],
    "comprad": ["en un radio de DIS", "a DIS de mi", "a DIS de donde estoy", "dentro de DIS", "a DIS de mi ubicación"],
    "compantind": ["cerca de mi", "mas cercano", "mas cerca de mi", "mas proximo", "mas proximo de mi"],
    "compantpl": ["cercanos a mi", "cerca de mi", "mas cercanos", "mas cerca de mi", "mas proximos", "mas proximos de mi"],
    "complug": ["en LOC", "en MUN", "dentro de LOC", "dentro de MUN", "en la colonia LOC"]
}

```

Figura 4.2.1. Diccionario de palabras que se utilizaron para formar oraciones con miles de combinaciones distintas.

```

rules = {
    "RADIO": [
        ["pregpl", "negpl", "verbpl", "comprad"],
        ["comprad", "pregpl", "negpl", "verbpl"],
        ["pregplw", "negplw", "comprad"],
        ["comprad", "pregplw", "negplw"],
        ["pregplw", "negplwque", "verbpl", "comprad"],
        ["comprad", "pregplw", "negplwque", "verbpl"]
    ],
    "CANTIDAD": [
        ["pregind", "negcantind", "verbind", "compcantind"],
        ["pregindw", "negcantindw", "compcantind"],
        ["pregindw", "negcantindwque", "verbind", "compcantind"],
        ["pregpl", "negcantpl", "verbpl", "compcantpl"],
        ["pregplw", "negcantplw", "compcantpl"],
        ["pregplw", "negcantplwque", "verbpl", "compcantpl"]
    ],
    "LUGAR": [
        ["pregpl", "negpl", "verbpl", "complug"],
        ["complug", "pregpl", "negpl", "verbpl"],
        ["pregplw", "negplw", "complug"],
        ["complug", "pregplw", "negplw"],
        ["pregplw", "negplwque", "verbpl", "complug"],
        ["complug", "pregplw", "negplwque", "verbpl"]
    ]
}

```

Figura 4.2.2. Reglas utilizadas para la combinación de palabras que generarían los datos de entrenamiento.

```

def createTrainData(name):
    cant_cycles = 0
    for rule in rules[name]:
        curr_cycles = 1
        quantity = []
        cycles_desc = []
        for tag in rule:
            quantity.append(len(dictionary[tag]))
        for i in range(len(quantity)-1, -1, -1):
            curr_cycles *= quantity[i]
            cycles_desc.insert(0, curr_cycles)
        cycles_desc.append(1)
        cant_cycles += curr_cycles

        for i in range(0, curr_cycles):
            if name=="CANTIDAD":
                updateRand()
            n = i
            indexes = []
            for q in cycles_desc:
                indexes.append(n//q)
                if n//q > 0:
                    n = n%q
            indexes.pop(0)
            j = 0
            sentence = ""
            while (j < (len(indexes))-1):
                sentence += dictionary[rule[j]][indexes[j]] + " "
                j+=1
            sentence += dictionary[rule[j]][indexes[j]] + ", " + name
            appendTxt('trainingData', sentence)

        print(str(cant_cycles), name, "sentences generated.")

```

Figura 4.2.3. Función que a partir de las palabras y reglas dadas genera de forma automática las oraciones que fueron utilizadas para el entrenamiento del modelo.

La combinación de palabras resultó en la generación automática de 8358 oraciones: 3718 con categoría/intención de *cantidad*, 2320 de *radio* y 2320 de *lugar*. Dichas oraciones fueron guardadas en un archivo .txt que posteriormente sería leído para el entrenamiento del modelo.

```
3718 CANTIDAD sentences generated.  
2320 RADIO sentences generated.  
2320 LUGAR sentences generated.
```

Figura 4.2.4. Cantidad de oraciones generadas que se utilizaron para el entrenamiento del modelo.

Estas oraciones o datos de entrenamiento se dividieron de forma aleatoria en 2 subsets: 80% para entrenamiento y 20% para pruebas. No está de menos mencionar que se descartó el uso de un 3er subset para validación ya que no se encontraron features de spaCy relacionadas a la validación del modelo durante el entrenamiento.

```
RADIO subsets: train - 1856 | test - 464  
CANTIDAD subsets: train - 2974 | test - 744  
LUGAR subsets: train - 1856 | test - 464  
  
Subsets Length: train - 6686 | test - 1672
```

Figura 4.2.5. División de datos de entrenamiento y pruebas.

```
nlp > training > trainingData.txt  
1741 que 80 negocios que ofrecen servicios de ACT se ubican mas cercanos, CANTIDAD  
1742 que 281 negocios que ofrecen servicios de ACT se ubican mas cerca de mi, CANTIDAD  
1743 que 130 negocios que ofrecen servicios de ACT se ubican mas proximos, CANTIDAD  
1744 que 15 negocios que ofrecen servicios de ACT se ubican mas proximos de mi, CANTIDAD  
1745 que 194 negocios que ofrecen servicios de ACT existen cercanos a mi, CANTIDAD  
1746 que 109 negocios que ofrecen servicios de ACT existen cerca de mi, CANTIDAD  
1747 que 245 negocios que ofrecen servicios de ACT existen mas cercanos, CANTIDAD  
1748 que 3 negocios que ofrecen servicios de ACT existen mas cerca de mi, CANTIDAD  
1749 que 31 negocios que ofrecen servicios de ACT existen mas proximos, CANTIDAD  
1750 que 13 negocios que ofrecen servicios de ACT existen mas proximos de mi, CANTIDAD  
1751 que 358 negocios de ACT estan cercanos a mi, CANTIDAD  
1752 que 302 negocios de ACT estan cerca de mi, CANTIDAD  
1753 que 80 negocios de ACT estan mas cercanos, CANTIDAD  
1754 que 208 negocios de ACT estan mas cerca de mi, CANTIDAD  
1755 que 208 negocios de ACT estan mas proximos, CANTIDAD  
1756 que 188 negocios de ACT estan mas proximos de mi, CANTIDAD  
1757 que 443 negocios de ACT hay cercanos a mi, CANTIDAD  
1758 que 489 negocios de ACT hay cerca de mi, CANTIDAD  
1759 que 407 negocios de ACT hay mas cercanos, CANTIDAD  
1760 que 70 negocios de ACT hay mas cerca de mi, CANTIDAD  
1761 que 33 negocios de ACT hay mas proximos, CANTIDAD  
1762 que 238 negocios de ACT hay mas proximos de mi, CANTIDAD  
1763 que 294 negocios de ACT se encuentran cercanos a mi, CANTIDAD  
1764 que 194 negocios de ACT se encuentran cerca de mi, CANTIDAD  
1765 que 143 negocios de ACT se encuentran mas cercanos, CANTIDAD  
1766 que 500 negocios de ACT se encuentran mas cerca de mi, CANTIDAD  
1767 que 20 negocios de ACT se encuentran mas proximos, CANTIDAD  
1768 que 322 negocios de ACT se encuentran mas proximos de mi, CANTIDAD  
1769 que 119 negocios de ACT se ubican cercanos a mi, CANTIDAD  
1770 que 221 negocios de ACT se ubican cerca de mi, CANTIDAD  
1771 que 103 negocios de ACT se ubican mas cercanos, CANTIDAD
```

Figura 4.2.6. Muestra de algunas de las oraciones generadas para el entrenamiento.

5. Detalles de Entrenamiento

5.1. Modelo de Reconocimiento de Entidades Nombradas

Este modelo no requiere un entrenamiento, únicamente hace uso de un diccionario de reglas para comparar y encontrar patrones dentro de una oración. La siguiente imagen muestra a detalle el formato de las reglas:

```
3 {"label":"DIS","pattern":[{"LIKE_NUM":true},{"TEXT":"metros"}]}
4 {"label":"DIS","pattern":[{"LIKE_NUM":true},{"TEXT":"mts"}]}
5 {"label":"LOC","pattern":"san rafael"}
6 {"label":"LOC","pattern":"santa cecilia"}
7 {"label":"LOC","pattern":"fraccionamiento del norte"}
```

Figura 5.1.1. Muestra del Diccionario de Patrones de Reconocimiento.

5.2. Modelo de Clasificación de Intenciones

SpaCy genera un documento con los detalles del modelo para su entrenamiento y permite la modificación de una gran cantidad de variables. A continuación se muestran los valores de algunas de las configuraciones más importantes y estudiadas durante el curso.

```
[training.optimizer]
@optimizers = "Adam.v1"
beta1 = 0.9
beta2 = 0.999
L2_is_weight_decay = true
L2 = 0.01
grad_clip = 1.0
use_averages = false
eps = 0.00000001
learn_rate = 0.001

[components.textcat_multilabel]
@architectures = "spacy.MaxoutSeqVecWrapper"
width = 64
window_size = 1
maxout_pieces = 3
depth = 2

[training.batcher.size]
@schedules = "compound"
start = 100
stop = 1000
compound = 1.001
t = 0.0
```

Figura 5.2.1. Parámetros significativos del modelo de clasificación de intenciones

Se pueden observar algunos valores tales como el uso del optimizador Adam con un learning rate de 0.001, con betha de .9; el uso de 2 capas de convolución (depth) con entradas y salidas de 64 de anchura, así como valores de batch inicial en 100 y finalización en 1000.

Respecto a los epoch de entrenamiento, se realizan 5 iteraciones.

```

---- TextCategorizer model training iteration 3 / 5 ... ----
100%|██████████| 1029/1029 [00:37<00:00, 27.24it/s]
Completed Iteration 3 - Loss: 10.394388536858855

---- TextCategorizer model training iteration 4 / 5 ... ----
100%|██████████| 1029/1029 [00:36<00:00, 28.38it/s]
Completed Iteration 4 - Loss: 6.792738748489644

---- TextCategorizer model training iteration 5 / 5 ... ----
100%|██████████| 1029/1029 [00:36<00:00, 27.99it/s]
Completed Iteration 5 - Loss: 4.98088587278398

Saving model...

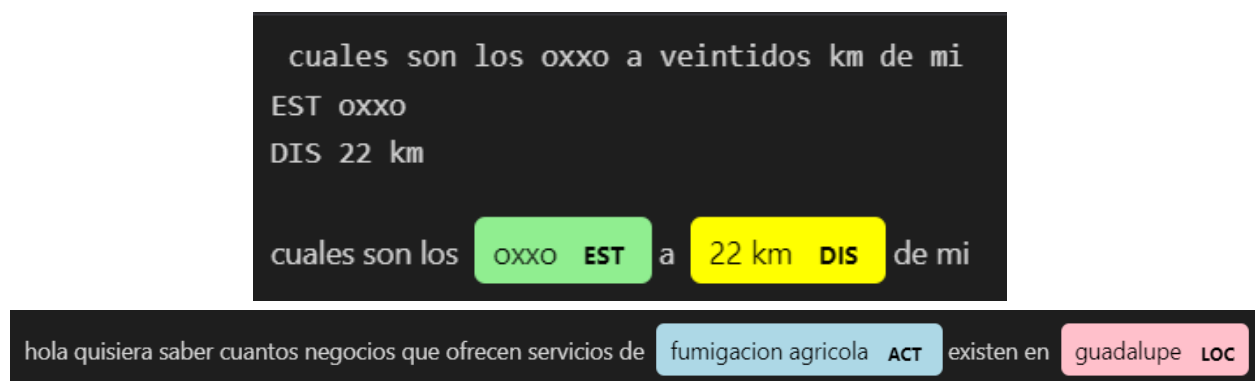
```

Figura 5.2.2. Iteraciones de entrenamiento del modelo de clasificación de intenciones.

6. Salidas de los Modelos

6.1 Modelo de Reconocimiento de Entidades Nombradas

En la *Figura 6.1.1. Ejemplo de Salidas del Modelo Entity Ruler* se observan los resultados del procesamiento de las oraciones “¿Cuáles son los oxxos a veintidós km de mi?” y “Hola, quisiera saber cuantos negocios que ofrecen servicios de fumigación agrícola existen en Guadalupe”; en la salidas se aprecia el preprocesamiento de dicha oración el cual se encargó de eliminar los acentos, signos de admiración, puntuación y conversión de mayusculas a minusculas para hacer el análisis del lenguaje más sencillo, se aprecia además el resultado final, donde se muestra que se reconoció la entidad de EST (establecimiento) oxxo y la entidad DIS (distancia) 22 km para el caso de la primera oración y, las entidades ACT (actividad) fumigacion agricola y LOC (Localidad/Colonia) guadalupe en la segunda oración.



```

cuales son los oxxo a veintidos km de mi
EST oxxo
DIS 22 km

cuales son los oxxo EST a 22 km DIS de mi

hola quisiera saber cuantos negocios que ofrecen servicios de fumigacion agricola ACT existen en guadalupe LOC

```

Figura 6.1.1. Ejemplos de salidas del modelo EntityRuler.

La función de este modelo reside en encontrar las variables que hay que buscar dentro de la base de datos del INEGI para generar una respuesta en base a ellas, por ejemplo, en la primer oración de la figura anterior se detecta que hay que buscar todos los registros del establecimiento Oxxo alrededor de un radio de 22 Km respectivo al usuario; Oxxo es la palabra clave a buscar mientras que la distancia se emplea para calcular la cantidad de registros a regresar.

La detección de entidades funge además como un auxiliar para la clasificación y detección de intenciones, a través de la existencia de ciertas entidades específicas dentro de la oración es posible descartar y seleccionar posibles intenciones del usuario, con ello, se facilita el análisis de la oración y se mejora la precisión del modelo de detección de intenciones.

6.2. Modelo de Clasificación de Intenciones

En la *Figura 6.2.1.* y *Figura 6.2.2.* se observan los resultados del procesamiento de las oraciones “Que tal, me gustaria que me mostraras los 5 negocios de abarrotes más cercanos a mí” y “Hola, quisiera saber cuantos negocios que ofrecen servicios de fumigación agrícola existen en Guadalupe”; en las imágenes se aprecia nuevamente el preprocesamiento del texto para facilitar su análisis así como los resultados arrojados por el modelo anteriores, se observa que el reconocimiento de entidades ya fue hecho; en la salida se muestra la probabilidad de que la oración pertenezca a cada una de las intenciones posibles. Para el primer caso la detección fue intención tipo “Cantidad” con una confianza del 99%, por su parte, el segundo caso detectó una intención de tipo “Radio” con una confianza del 100%.

```
que tal me gustaria que mostraras los 5 negocios de ACT mas cercanos  
{ 'RADIO': 2.740822235836049e-09, 'CANTIDAD': 0.9999994039535522, 'LUGAR': 2.0301098629715852e-05 }
```

Figura 6.2.1. Ejemplo del modelo Text Categorizer para clasificación de la intención de tipo “cantidad”.

```
hola quisiera saber cuantos negocios que ofrecen servicios de ACT existen en LOC  
{ 'RADIO': 2.866839876602967e-09, 'CANTIDAD': 5.108543064125115e-06, 'LUGAR': 1.0 }
```

Figura 6.2.2. Ejemplo del modelo Text Categorizer para clasificación de la intención de tipo “radio”.

La función de este modelo reside en encontrar qué tipo de búsqueda y extracción de datos se debe de realizar de la base de datos; existen diferentes algoritmos de obtención de información

dependientes a las necesidades de la intención del usuario, por ejemplo: en el caso de que la búsqueda sea por radio es necesario realizar una conversión de las latitudes a kilómetros, o en el caso de que la intención sea por lugar, hay que utilizar la columna pertinente a localidad o municipio en la base de datos para filtrar los resultados.

7. Comparación y Evaluación de los Modelos

7.1 Métricas de Evaluación

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

- Precisión / Accuracy
- 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.

7.2. Modelo de Reconocimiento de Entidades Especiales

7.2.1. Modelo Entity Ruler vs Modelo NER

Como se mencionó previamente, el modelo EntityRuler 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, en un principio se intentó utilizar el modelo NER (Named Entity Recognizer), el cual si lleva un entrenamiento por iteraciones donde se implementa un algoritmo de deep learning pero se requiere de una inmensa cantidad de oraciones donde se ejemplifique la presencia de las entidades a reconocer. El uso de este modelo habría requerido 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 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.

7.2.2. 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 7.2.2.1. 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 7.2.2.1. *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 7.2.2.2. 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).

```
nlp > ruler_model > entity_ruler > {} patterns.jsonl
1 {"label":"DIS","pattern":[{"LIKE_NUM":true},{"TEXT":"kilometros"}]}
2 {"label":"DIS","pattern":[{"LIKE_NUM":true},{"TEXT":"km"}]}
3 {"label":"DIS","pattern":[{"LIKE_NUM":true},{"TEXT":"metros"}]}
4 {"label":"DIS","pattern":[{"LIKE_NUM":true},{"TEXT":"mts"}]}
5 {"label":"LOC","pattern":"san rafael"}
6 {"label":"LOC","pattern":"santa cecilia"}
7 {"label":"LOC","pattern":"fraccionamiento del norte"}
8 {"label":"LOC","pattern":"del prado"}
9 {"label":"LOC","pattern":"infontavit solidaridad"}
10 {"label":"LOC","pattern":"fabriles"}
11 {"label":"LOC","pattern":"san vicente"}
12 {"label":"LOC","pattern":"jicacal"}
13 {"label":"LOC","pattern":"las huertas"}
14 {"label":"LOC","pattern":"rayones"}
15 {"label":"LOC","pattern":"loma chula"}
16 {"label":"LOC","pattern":"predio aldape"}
17 {"label":"LOC","pattern":"lagrange"}
18 {"label":"LOC","pattern":"colinas de san jeronimo"}
19 {"label":"LOC","pattern":"garcia mireles"}
```

Figura 7.2.2.2. Diccionario de patrones de reconocimiento.

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.

7.3. Modelo de Clasificación de Intenciones

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

En un inicio se intentó realizar la clasificación/categorización de intenciones 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 utilizando el módulo de TextCategorizer de la librería de spaCy.

```
cuales son los EST a DIS de mi  
{ 'RADIO': 1.0, 'CANTIDAD': 1.1298060599074233e-05, 'LUGAR': 6.520797910525289e-07}
```

Figura 7.3.1.1. 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 u  
bicación","mts de donde estoy"],"LUGAR":["en","dentro de"]},"input":"es","output":"es","knowledge_base_origin":"database"},"result":{"sentence":"Que oxos hay a 5 kilómetros  
de mí","equivalent":"Que oxos hay a 5 kilómetros de mí","categories":{"LUGAR":0.33572155237197876,"RADIO":0.20143739879131317,"CANTIDAD":0.000017619813661440276}}}
```

Figura 7.3.1.2. 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 7.3.1.3 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 7.3.1.3. Historial de loss durante las iteraciones para el entrenamiento del modelo TextCategorizer.

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 7.3.1.3. 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.