Soluciones a la Práctica 1

Minería de Textos

Sabela Alicia Iglesias Sánchez

Noviembre 2024

1. Tarea CoNLL2002

La tarea colaborativa **CoNLL-2002** se centró en el Reconocimiento de entidades nombradas (NER) en los idiomas Español y Neerlandés. Su objetivo fue proponer distintos sistemas de NER multilingües. Es decir, procesos que desempeñen correctamente la tarea NER para varios idiomas.

Se proporcionaron datos de entrenamiento y test estructurados en formato IOB2 o IOBES que constaban de cuatro entidades principales: PER (persona), LOC (localización), ORG (organización) y MISC (miscelánea). Además, los datos estaban en dos lenguas: español y neerlandés. Los datos en español constaban de una colección de artículos de noticias. Los datos en neerlandés consistieron en cuatro ediciones del periódico Belga "De Morgen".

El sistema baseline propuesto para esta tarea identificaba unicamente las entidades con una clase única en los datos de entrenamiento y, si la frase estaba contenido en más de una entidad, se elegía la de mayor longitud.

Los participantes presentaron doce sistemas distintos para la realización de la tarea. Estos sistemas se basaron, principalmente, en los siguientes métodos: Support Vector Machine, Decission Trees, Transformation-based Learning, Algoritmos AdaBoost, Modelo Hidden Markov, Maximun entropy models y Memory-based Learning.

Estos sistemas se evaluaron sobre los mismos datos de test utilizando para ello la medida F_β para $\beta = 1$. Es decir, la $F_{\beta=1} = \frac{2 \cdot p \cdot r}{p+r}$ donde las métricas p y r son, respectivamente, los valores de *precision*(porcentaje de entidades correctas de las encontradas por el sistema) *recall* (porcentaje de entidades encontradas) obtenidos.

Todos los sistemas presentados superaron al baseline. Los mejores resultados en las tres métricas (*precision*, *recall* y $F_\beta = 1$) fueron obtenidos por Carreras, Márquez y Padró que obtuvieron los valores $p = 81{,}38\,\%$ $r = 81{,}40\,\%$ $F_{\beta=1} = 81{,}39$ para la lengua española y $p = 77{,}83\,\%$ $r = 76{,}29\,\%$ $F_{\beta=1} = 77{,}05$ para la neerlandesa. El sistema presentado se basó en árboles de decisión de profundidad fija y el algoritmo AdaBoost. Este sistema combinaba clasificadores débiles (árboles de decisión) para conseguir un clasificador fuerte. Este enfoque de árboles de decisión utilizó características como: información contextual, pistas presentes en las palabras, entidades previamente conocidas, listas de palabras externas y etiquetas POS. Además, este procedimiento se realizó en dos etapas: una primera etapa de reconocimiento de entidades y una segunda etapa de clasificación de estas.

El sistema presentado por Wu, Ngai, Carpuat, Larsen and Yang tuvo un enfoque similar al anterior, utilizando el algoritmo AdaBoostMH y árboles de decisión de profundidad fija 1. El desempeño de este sistema fue algo menor que el de Carreras en ambos idiomas, para el español se obtuvo $F_{\beta=1} = 76{,}61$ y para el neerlandés $F_{\beta=1} = 75{,}36$. Sin embargo, también se obtuvieron buenos resultados independientemente del idioma.

Por último destacaremos el sistema propuesto por Florian, los valores $F_{\beta=1} = 79{,}05$ en español y $F_{\beta=1} = 74{,}99$. Este sistema utilizó aprendizaje métodos de aprendizaje automático basados en transformaciones para realizar la identificación de entidades y Snow para mejorar estas identificaciones previas. Para la etapa de categorización se llevó a cabo un algoritmo de *forward-backward*.

2. Etiquetador de entidades nombradas y corpus de test conll2002

El objetivo de esta sección es utilizar el etiquetador de entidades nombradas de Spacy (utilizando el modelo preentranado en español) sobre los datos del corpus conll2002. El archivo esp.testb.txt contiene textos etiquetados de noticias en español en el formato IOB. Añadimos aquí las primeas líneas del archivo a modo explicativo:

```
La B-LOC
Coruña I-LOC
, O
23 O
may O
( O
EFECOM B-ORG
) O
. O

- O

Las O
reservas O
```

2.1. Recuperar texto plano

Para procesar el texto con el modelo preentrenado de Spacy es necesario obtener el texto plano a partir del archivo etiquetado esp.testb.txt. Se ha creado la función extract_plain_text para recuperar el texto plano de un archivo etiquetado en formato IOB.

```
def extract_plain_text(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        lines = file.readlines()

    text = []
    for line in lines:
        if line.strip():
            word = line.split()[0]  # Toma la primera columna (palabra)
            text.append(word)
        else:
            text.append('')  # Si la linea esta vacia es espacio

    # Unimos las palabras con espacios
    plain_text = '␣'.join(text)
    return plain_text
```

Procesamos el texto con esta función y guardamos un archivo plain_text.txt.

```
plain_text = extract_plain_text('TESTB-ESP.txt')
with open('plain_text.txt', 'w', encoding='utf-8') as out_file:
    out_file.write(plain_text)
```

2.2. Procesamiento del texto

En esta sección utilizaremos el modelo preentrenado de procesamiento de lenguaje natural de spacy en español para generar etiquetas en formato IOB para la tarea NER. Se ha cargado el modelo "es_core_news_sm", un modelo entrenado con noticias en castellano. Una vez cargado este modelo, es necesaria ajustar la tokenización, es decir, las reglas que utiliza el modelo para dividir el texto el porciones que luego clasificará. Es importarte elegir la tokenización adecuada para que los resultados obtenidos sean comparables con el archivo esp.testb.txt.

```
##Cargamos el modelo preentrenado y customizamos la tokenizacion
import spacy
from spacy.tokenizer import Tokenizer
nlp = spacy.load("es_core_news_sm")
nlp.tokenizer=Tokenizer(nlp.vocab)

# Leer el texto desde un archivo
```

```
with open("plain_text.txt", "r", encoding="utf-8") as file:
    text = file.read()

doc = nlp(text)
```

La salida de este modelo es una estructura que representa el texto procesado. Para poder compararlo con los valores etiquetados reales se ha creado función text_to_iob que procesa cada token de la salida del modelo de procesamiento de lenguaje natural y escribe una línea con la entidad y su etiqueta IOB.

La función text_to_iob nos permite reformatear el output al un formato idéntico al del archivo test.espb.txt. Sin embargo, para poder evaluar el output de nuestro modelo con el entorno conlleval.py es necesario crear un archivo con el formato entidad-true label - predicted label. Esto se ha tratado a través de la función combine.

```
    def text_to_iob(doc):
    iob_output = []
    for token in doc:
        # Si el token esta dentro de una entidad nombrada (B, I, O)
        if token.ent_iob_ == "B":
            iob_output.append(f"{token.text} B-{token.ent_type_}")
        elif token.ent_iob_ == "I":
            iob_output.append(f"{token.text} I-{token.ent_type_}")
        elif token.text!=' ':
            # Si no esta dentro de una entidad nombrada, es Outside (O)
            iob_output.append(f"{token.text} O")
        else:
            iob_output.append(' ')

    iob_output = "\n".join(iob_output)

    return iob_output
```

```
    def combine (true_labels_file, pred_labels_file, output_file):
    """
    Combina las etiquetas reales y predichas en un unico archivo en formato CoNLL.

    Args:
        true_labels_file (str): Ruta del archivo con las etiquetas reales.
        pred_labels_file (str): Ruta del archivo con las etiquetas predichas.
        output_file (str): Ruta del archivo de salida combinado.

    Returns:
        None
    """
    # Leemos ambos archivos
    with open(true_labels_file, "r", encoding="utf-8") as f:
        true_lines = f.read().strip().splitlines()

    with open(pred_labels_file, "r", encoding="utf-8") as f:
        pred_lines = f.read().strip().splitlines()

    # Creamos el archivo combinado
    with open(output_file, "w", encoding="utf-8") as f:
        for true_line, pred_line in zip(true_lines, pred_lines):
            if true_line.strip() == "" or pred_line.strip() == "":
                # Linea vacia (separaci n de oraciones)
                f.write("\n")
                continue

            # Dividir palabras y etiquetas
            true_word, true_label = true_line.rsplit(maxsplit=1)
            pred_word, pred_label = pred_line.rsplit(maxsplit=1)

            f.write(f"{true_word} {true_label} {pred_label}\n")

        f.write("\n")
```

Procesando el output de del modelo con las dos funciones anteriores hemos creado el archivo combined.txt.

```
spacy_otput=text_to_iob(doc)
```

```
with open('pred_labels.txt', 'w', encoding='utf-8') as out_file:
    out_file.write(spacy_otput)
# Nombres de los archivos
true_labels_file = "TESTB-ESP.txt"
pred_labels_file = "pred_labels.txt"
output_file = "combined.txt"
combine(true_labels_file,pred_labels_file,output_file)
```

Mostramos las primeras líneas del archivo combine.txt:

```
La B-LOC O
Coruña I-LOC B-MISC
, O O
23 O O
may O O
( O O
EFECOM B-ORG B-MISC
) O O
. O O

- O O

Las O O
reservas O O
```

2.3. Evaluación de los resultados obtenidos

Para evaluar el desempeño del modelo nos basaremos en tres métricas: precisión, recall y F_1. Estas métricas se definen como:

$$precisión = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN} \qquad F_1 = 2 \cdot \frac{precisión \cdot recall}{precisión + recall}$$

donde TP=true positive, TN=true negative y FN=false negative. Las definiciones anteriores se interpretan como sigue:

La precisón es el porcentaje de entidades correctamente etiquetadas de entre toda las entidades etiquetadas.

El recall es el porcentaje de entidades etiquetadas correctamente de entre todas las entidades que verdaderamente corresponden a la etiqueta.

La métrica F_1 es la media armónica entre precisión y recall, balancea ambas medidas y nos da una noción general del desempeño del modelo.

Sin embargo, estas definiciones son útiles en un cotexto binario (true-false) donde solo existe una etiqueta y esta se asigna o no a cada individuo. En nuestro problema NER tenemos cuatro etiquetas: PER, ORG, LOC, MISC. Para adaptar los conceptos anteriores a contextos multi-etiqueta utilizaremos el entorno conlleval.py proporcionado. Este entorno sigue una estrategia basada en chunks (fragmentos del texto etiquetados correcta o incorrectamente, estos fragmentos tienen su inicio y fin marcados por los marcadores B y I) para el cálculo del as métricas. Utilizando la función evaluate_conll_file obtendremos los valores por categoría de estas tres medidas y valores globales que se han obtenido como sigue:

1. En primer lugar realiza un conteo de chunks, y almacena el número de chunks correctamente etiquetados, el número verdadero de chunks por categoría y el número de chunks predichos por categoría.
2. Calcula los valores de $precisión_C$, $recall_C$ y F_{1C} de cada categoría como:

$$precisión_C = \frac{[correct - chunks]_C}{[pred - chuncks]_C} \qquad recall_C = \frac{[correct - chunks]_C}{[true - chuncks]_C}$$

$$F_{1C} = 2 \cdot \frac{precisión_C \cdot recall_C}{precisión_C + recall_C}$$

3. Los valores Globales se calculan con un enfoque micropromedio, contabilizando $TP = \sum_C TP_C$, $TN = \sum_C TN_C$ y $TN = \sum_C TN_C$ a través de todas las categorías. Con estos valores ya es posible calcular las métricas con las definiciones dadas al inicio de esta sección. Este enfoque micropromedio otorga más peso a aquellas categorías con mayor representación en el texto lo que puede ocultar un bajo rendimiento del modelo en la clases menos representadas, sin embargo, es útil para conocer el desempeño general.

Vamos a interpretar ahora los resultados de nuestro modelo:

```
# Importar el script
import conlleval
from conlleval import evaluate_conll_file

with open("combined.txt", "r") as file:
    result = evaluate_conll_file(file)

print(result)
```

Output:

```
    processed 53050 tokens with 3559 phrases; found: 3911 phrases; correct: 2027.
accuracy:  56.07%; (non-O)
accuracy:  92.38%; precision:  51.83%; recall:  56.95%; FB1:  54.27
              LOC: precision:  50.82%; recall:  71.68%; FB1:  59.47  1529
             MISC: precision:  12.46%; recall:  21.76%; FB1:  15.85  594
              ORG: precision:  72.98%; recall:  42.64%; FB1:  53.83  818
              PER: precision:  59.69%; recall:  78.78%; FB1:  67.92  970
(51.82817693684479, 56.95420061815116, 54.270414993306545)
```

De estos resultados destacamos tres métricas globales **precision=51.83 %**, **recall=56.95**, $F_{\beta=1} = 54{,}27$.

La precisión de un modelo de NER se calcula como el porcentaje de las entidades categorizadas correctamente sobre el número total de entidades en el texto. En este caso hemos obtenido una precisión del 51,83 %.

El recall es 56,95 %. Esto significa que de todas las entidades que realmente estaban en el texto (es decir, las entidades verdaderas), el modelo logró identificar aproximadamente el 56,95 %.

El $F_{\beta=1}$ es 54,27 %, que es la media armónica entre precisión y recall. Un F1 más alto indica un equilibrio adecuado entre precisión y recall. El F1 busca equilibrar la necesidad de ser preciso (evitar errores de clasificación) con la necesidad de encontrar todas las entidades posibles.

En general, los tres resultados anteriores indican un rendimiento moderado del modelo. Si nos fijamos en los valores de las métricas por categorías podemos observar que el modelo obtiene resultados ligeramente mejores en la tarea de clasificación dé la etiqueta PER y que los valores significativamente más pobres se han obtenido en la tarea de categorización con etiqueta MISC (la menos representada con 594 apariciones).

2.4. Propuestas para mejorar los resultados obtenidos

Existen multitud de aproximaciones a este problema que pueden ayudar a mejorar el rendimiento del modelo, listaremos y explicaremos tres de ellas:

2.4.1. Agregar diccionarios con datos específicos

El modelo preentrenado para NER de Spacy es una herramienta útil para la categorización de entidades en general, sin embargo, cuando los textos tratan de un tema en particular con léxico especifico (noticias deportivas, legales, políticas) el desempeño puede empeorar. Esta casuística puede tratarse con un enfoque fine-tune, añadiendo diccionarios específicos correctamente categorizados a los datos del modelo. De este modo, el modelo general de spacý mejorará su desempeño en la categorización de entidades concretas. Este enfoque también puede realizar por categorías, añadiendo, por ejemplo, un diccionario de palabras que el modelo debe categorizar como PER o LOC.

2.4.2. Pipeline de desambiguación léxica

Un problema común para los sistemas de reconocimiento de lenguaje natural es la ambigüedad léxica, es decir, la ambigüedad que se das cuando una sola palabra representa distintas entidades dependiendo del contexto (por ejemplo, la palabra Barcelona es una LOC pero si viene acompañada de FC Barcelona debe ser etiquetada como ORG). Para que los modelos puedan interpretar y reconocer mejor este tipo de entidades es necesario implementar algún proceso de desambiguación. Normalmente, estos procesos tratan de descifrar la etiqueta real de la entidad ambigua basándose en su contexto.

En el caso que nos ocupa es posible implementar un pipeline al modelo que emplee técnicas de aprendizaje automático o analice el contexto semántico de la entidad.

2.4.3. Combinar el modelo con deep learning

Un enfoque un poco más complejo a los anteriores para mejorar los resultados de la clasificación es combinar este modelo con algoritmos de deep learning que han demostrado ser eficaces en tareas como el reconocimiento de entidades nombradas. Combinado los resultados obtenidos anteriormente con los resultados de los nuevos modelo mediante un modelo de ensamble es posible mejorar el desempeño con respecto a utilizar tan solo el algoritmo de Spacy.

3. Ejercicio opcional: Clasificación NER manual

Como parte de la parte opcional de esta práctica hemos seleccionado algunos párrafos de distintas noticias y los hemos clasificado manualmente.

Así está la clasificación de LaLiga EA Sports, en directo.

La Real Sociedad superó este domingo al Leganés (0-3), mientras que el Athletic derrotó al Villarreal (0-2), un rival directo en la lucha por los puestos de Champions La jornada 16ª arrancó el viernes con un triunfo (2-0) del Celta ante el Mallorca y el sábado, la UD Las Palmas venció (2-1) al Valladolid, el Betis empató ante el Barça (2-2) y el Valencia cayó (0-1) contra el Rayo. El Girona-Real Madrid se saldó con un claro 0-3 para los merengues Partido entre Leganés y Real Sociedad.

En la jornada de este domingo, la Real Sociedad ha prolongado su buena racha para sumar su cuarta victoria consecutiva y meterse en zona europea. Los de Imanol Alguacil se impusieron por 0-3 al Leganés con los goles de Brais Méndez en el 14', Barrenetxea en el 78' y Oyarzábal en el 91'. Por su parte, el Athletic sigue con su fenomenal racha y se impuso con goles de Paredes e Iñaki Williams al Villarreal (2-0), un rival directo en la lucha por los puestos de Champions.

Superior de Justicia de Madrid (TSJM) ha devuelto la causa abierta contra Ana Millán, vicepresidenta de la Asamblea regional y número tres del PP de Madrid, a la jueza de Navalcarnero para que complete la investigación y, en su caso, remita una exposición razonada con la imputación, ya que por ahora ha expuesto ünа serie de hechos que objetivamente no ostentan por sí la naturaleza de delito".
En el auto, la Sala de lo Civil y lo Penal del TSJM rechaza la inhibición que planteó el Juzgado de Primera Instancia e Instrucción número 6 de Navalcarnero el pasado junio (en la que argumentó que Millán es aforada), le devuelve las actuaciones .en este momentoœ cierra las diligencias que abrió el TSJM en julio.
Los magistrados concluyen que, como sostuvo la Fiscalía, la instrucción .está inconclusa, en relación a la práctica de diligencias acordadas y no practicadas y que se califican de esenciales", y además hay pendientes recursos de apelación.

Asistimos hace una semana, en Inca, a la I Fira de la Vida i la Mort, montada por 'Una vida ONG', entidad creada por Daniela Rueda en 2013, a su vez fundadora, y propietaria, de 'Una vida Mallorca SL', y de la primera 'Funeral Planner de Baleares', por lo cual ha sido subvencionada por el Consell de Mallorca como mujer emprendedora con un proyecto empresarial singular que contribuye a la transformación del modelo económico y social... Y nunca mejor dicho lo de singular, puesto que antes de que ella apareciera, «lo que se hacía respecto a funerales, tanto laicos como religiosos, nada tienen que ver con los que organiza, en los que por encima de todo se siente el amor, pues la muerte es algo que inevitablemente llegará, por tanto, aprender de ello, ayudará a afrontarla mejor. Lo importante es ritualizar ese acto tan importante que no debe caer en el olvido. Me inicié en esto con el funeral que le hice a mi madre, con la música que a ella le gustaba, tocada en directo, con asistencia de la gente a la que quería y que la quería. Lo mismo sucedió hace unas semanas, cuando falleció mi padre, una persona a la que le gustaba la fiesta, tanto que a dónde él llegaba se terminaba la tristeza. Por eso, su funeral fue un tributo a esta, como a él le hubiera gustado. De hecho, desde que tengo Una vida Mallorca son muchos los homenajes fúnebres y acompañamiento en el duelo que he realizado».

Procesamos estos tres textos almacenados como texto plano en el documento texto_manual_plano.txt. Los valores de la categorización manual se han almacenado en el documento true_labels2.txt. Procesamos estos documentos de igual manera que hemos hecho en la sección anterior e imprimimos el resutlado de la evaluación.

```
#Cargamos el modelo nlp
nlp = spacy.load("es_core_news_sm")
nlp.tokenizer=Tokenizer(nlp.vocab)
with open("texto_manual_plano.txt", "r", encoding="utf-8") as file:
    text3 = file.read()

# Procesar el texto con SpaCy
doc3 = nlp(text3)
#Guardamos la clasificaci n predicha enun documento pred_labels3.txt
spacy_output3=text_to_iob(doc3)
with open('pred_labels3.txt', 'w', encoding='utf-8') as out_file:
    out_file.write(spacy_output3)

# Creamos el documento en formato word gold pred
true_labels_file = "true_labels3.txt"
pred_labels_file = "pred_labels3.txt"
output_file = "combined3.txt"
combine(true_labels_file,pred_labels_file,output_file)

#Sacamos las m tricas para evaluar el resultado final
with open("combined3.txt", "r") as file:
    result = evaluate_conll_file(file)

print(result)
```

Output:

```
processed 689 tokens with 50 phrases; found: 60 phrases; correct: 6.
accuracy:  16.83%; (non-O)
accuracy:  77.21%; precision:  10.00%; recall:  12.00%; FB1:  10.91
              LOC: precision:   9.09%; recall:  12.50%; FB1:  10.53  11
             MISC: precision:  12.50%; recall:  60.00%; FB1:  20.69  24
              ORG: precision:  12.50%; recall:   7.14%; FB1:   9.09  16
              PER: precision:   0.00%; recall:   0.00%; FB1:   0.00  9
(10.0, 12.0, 10.90909090909091)
```

Los resultados del modelo muestran que tiene un desempeño bastante bajo en la tarea de etiquetado de entidades. La precisión global es de solo un 10 %, con un recall del 12 % y un $F_1 = 10{,}91$. Lo más destacable es que no ha sido capaz de reconocer ninguna entidad de tipo **persona (PER)**, lo cual es una falla importante. Aunque su rendimiento en **ubicaciones (LOC)** y **misceláneos (MISC)** es ligeramente mejor, sigue siendo insuficiente. En cuanto a **organizaciones (ORG)**, el modelo también tiene problemas, con una precisión de 12,5 % y un recuerdo de 7,14 %. Estos resultados son significativamente peores que los encontrados en la primera sección.