Universidad Nacional
de Educación a Distancia

Escuela Técnica Superior de Ingeniería Informática

Práctica 1: Detección de Entidades Nombradas

Autora: Sabela Alicia Iglesias Sánchez

Minería de Textos

Máster Universitario
en Ciencia e Ingeniería de Datos

19 de noviembre de 2025

Índice

Todo el material asociado a esta práctica (código y ficheros generados) está disponible en un repositorio público de GitHub: https://github.com/sabelaalicia/conll2002-mt-practica1.git

1. Tarea CoNLL2002

1.1. Descripción de la tarea

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 resultados obtenidos por los distintos participantes 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) y *recall* (porcentaje de entidades encontradas) obtenidos.

1.2. Sistemas propuestos por los participantes

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.

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*.

1.3. Resultados de la tarea

Los resultados de los participantes mostraron un desempeño sólido en la tarea de NER. Sin embargo, más allá de los resultados individuales, se alcanzaron objetivos más esenciales del taller. Por un lado, se logró evaluar de forma comparativa el rendimiento de distintos sistemas, lo que impulsó la investigación en el campo en ese momento. Por otro lado, los resultados obtenidos permitieron establecer un marco de referencia útil para comparar futuros sistemas desarrollados en el ámbito del NER. De este

modo, se contribuyó al desarrollo de los sistemas NER por partida doble: fomentando la investigación en su momento y creando un marco de comparación para trabajos futuros.

Es importante notar que, aunque los resultados de CoNLL-2002 fueron relevantes, los sistemas dependían de características manuales y modelos como árboles de decisión o Máxima Entropía. Con la llegada de Transformers como BERT, se superaron estas limitaciones, aprendiendo representaciones contextuales y logrando rendimientos mucho más altos, dejando los métodos de CoNLL-2002 obsoletos para el momento actual.

2. Solución propia a la tarea

En esta sección se construirá una solución para la tarea de NER descrita, con el objetivo de comparar posteriormente los resultados obtenidos con el sistema de referencia proporcionado por CoNLL-2002. Para abordar esta tarea, se empleará el etiquetador de entidades nombradas en español disponible en la librería spaCy. Dado que se trata de un modelo preentrenado, no será necesario utilizar los datos de entrenamiento, sino que se procederá directamente al procesamiento de los datos de prueba contenidos en el archivo *esp.testb.txt*, el mismo conjunto de test empleado por los participantes para evaluar el rendimiento de sus sistemas en el idioma español.

2.1. Recuperar texto plano

El conjunto de datos de test están etiquetados en formato IOB (*Inside–Outside–Beginning*), un esquema de anotación en el cual se indica la posición de cada palabra respecto a una entidad dentro del texto. Cada token se etiqueta con una de las tres posibles marcas (`I`, `O`, `B`):

- `B`: el token es el comienzo de una entidad.
- `I`: el token forma parte de una entidad que comenzó anteriormente.
- `O`: el token no pertenece a ninguna entidad.

Podemos observar una pequeña muestra del documento real de datos de test etiquetado:

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

- O

Las O
reservas O
```

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.

```python
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('../data/raw/TESTB-ESP.txt')
with open('../data/processed/plain_text.txt', 'w', encoding='utf-8') as out_file:
    out_file.write(plain_text)
```

A continuación, se muestra el resultado de procesar las primeras lineas:

```
La Coruña , 23 may ( EFECOM ) . - Las reservas '' on line '' de billetes aéreos a través de Internet aumentaron en España un 300 por ciento
```

2.2. Procesamiento del texto

En esta sección se empleará el modelo en español de `spaCy` para generar etiquetas en formato IOB en la tarea NER. Para ello, se ha cargado el modelo `es_core_news_sm`, entrenado con textos periodísticos en castellano.

Este modelo utiliza un tokenizador basado en reglas lingüísticas y expresiones regulares, adaptadas al idioma español. Para poder comparar los resultados del modelo con los datos de referencia (*gold data*), es imprescindible que la tokenización sea idéntica en ambos casos. Por ello, se ha diseñado la función `extract_plain_text` de forma que se genere un texto en el que los tokens definidos en el *gold data* están separados por espacios en blanco. De este modo, es suficiente con sustituir el tokenizador por defecto del modelo por uno más sencillo, basado únicamente en la separación por espacios.

```
##Customizamos la tockenizaci n y cargamos un modelo preentrenado
nlp = spacy.load("es_core_news_sm")
nlp.tokenizer=Tokenizer(nlp.vocab)

# Leer el texto desde un archivo
with open("../data/processed/plain_text.txt", "r", encoding="utf-8") as file:
    text = file.read()

# Procesar el texto con SpaCy
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*.

```
    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
```

Sin embargo, para poder evaluar el output de nuestro modelo con el módulo `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 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 (separaciON 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 del modelo con las dos funciones anteriores hemos creado el archivo *combined.txt*.

```
spacy_otput=text_to_iob(doc)
with open('../data/processed/pred_labels.txt', 'w', encoding='utf-8') as out_file:
    out_file.write(spacy_otput)

# Nombres de los archivos
true_labels_file = "../data/raw/TESTB-ESP.txt"
pred_labels_file = "../data/processed/pred_labels.txt"
output_file = "../data/processed/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

2.3.1. Métricas

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

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

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 *precision* 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 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 módulo `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 de las 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 $precision_C$, $recall_C$ y F_{1C} de cada categoría como:

$$precision_C = \frac{[correct\_chunks]_C}{[pred\_chuncks]_C} \qquad recall_C = \frac{[correct\_chunks]_C}{[true\_chuncks]_C}$$

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

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 de un sistema.

2.3.2. Resultados

Vamos a interpretar ahora los resultados de nuestro modelo utilizando estas métricas.

```
# EVALUACION
with open("../data/processed/combined.txt", "r") as file:
    result = evaluate_conll_file(file)

print("\nRESULTADOS DE LA EVALUACION:")
print("=" * 50)
print(result)
print("=" * 50)
```

Output:

```
processed 53050 tokens with 3559 phrases; found: 3945 phrases; correct: 2047.
accuracy:  56.23%; (non-O)
accuracy:  92.23%; precision:  51.89%; recall:  57.52%; FB1:  54.56
```

```
              LOC: precision:  51.20%; recall:  71.03%; FB1:  59.51  1504
             MISC: precision:  12.36%; recall:  23.53%; FB1:  16.21  647
              ORG: precision:  75.88%; recall:  44.50%; FB1:  56.10  821
              PER: precision:  58.99%; recall:  78.10%; FB1:  67.21  973

RESULTADOS DE LA EVALUACIÓN:
================================================
(51.88846641318124, 57.516156223658335, 54.55756929637528)
================================================
```

De estos resultados destacamos tres métricas globales ***precision*=51.89 %**, ***recall*=57.52 %**, $F_{\beta=1}$**=54.56 %**.

La *precision* 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 *precision* del 51,89 %.

El *recall* es 57,52 %. Esto significa que de todas las entidades que realmente estaban en el texto (es decir, las entidades verdaderas), el modelo logró identificar aproximadamente el 57,52 %.

El $F_{\beta=1}$ es 54,56 %, que es la media armónica entre *precision* y *recall*. Un F_1 más alto indica un equilibrio adecuado entre *precision* y *recall*. El F_1 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 de 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 647 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 spacy 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 da 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 una noticia publicada en el periódico *El Mundo* el día 18.11.2025 accesible desde este enlace.

> *El refuerzo de la vacunación antigripal llega a Valladolid a partir del 28 de noviembre, sin necesidad de cita previa y con la posibilidad de acudir a recibir las dosis durante los fines de semana. Para garantizar la asistencia, la Consejería de Sanidad habilita 24 de Puntos de Atención Continuada (PAC) y centros de guardia en la provincia.*
>
> *La última actualización de la campaña de vacunación frente a los virus respiratorios por parte del departamento dirigido por Alejandro Vázquez llega tras una última semana en la que Castilla y León alcanzó el umbral epidémico en casos de gripe, con 51 casos por 100.000 habitantes. Los mayores contagios están repercutiendo en niños entre 5 y 14 años, si bien se ha observado un incremento de infecciones respiratorias en las personas mayores de 75 años.*
>
> Por ello, la Consejería de Sanidad anuncia que el próximo 29 de noviembre ya estará habilitada la posibilidad de vacunación los fines de semana y sin cita previa frente a la gripe y el Covid-19. No obstante, se mantendrá el horario de mañana de lunes a viernes para aquellos usuarios que quieren recibir sus correspondientes dosis en el centro de salud correspondiente, así como los profesionales sanitarios en su centro de trabajo.
>
> No solo Valladolid se nutre de este refuerzo antivirus respiratorias, sino que se traslada al resto de Castilla y León, de forma que serán 196 los PAC y centros de guardia autorizados para este proceso, repartidos de la siguiente forma: Ávila, 20; Burgos, 25; León, 23; El Bierzo, 8; Palencia, 16; Salamanca, 28; Segovia, 19; Soria, 14; Valladolid Oeste, 11; Valladolid Este, 13 y Zamora, 19.
>
> Los registros de Sanidad indican que ya se han vacunado de gripe un total de 545.037 personas, con una cobertura en mayores de 60 años del 48,7 %, por lo que se quieren hacer un especial llamamiento a este grupo de edad para que alcancen cuanto antes la inmunidad.
>
> En cuanto a la tasa de incidencia de síndrome gripal, los datos reflejan que se ha duplicado en la semana hasta los 130 casos por 100.000 habitantes, si bien continúa en nivel de intensidad bajo, aunque ya está por encima del umbral epidémico. En concreto, la tasa global de infecciones respiratorias agudas (IRA) se mantiene estable, con 621 casos por 100.000 habitantes, según el último informe de vigilancia epidemiológica de la Red Centinela Sanitaria de Castilla y León recogido por Ical.
>
> Aunque se ha producido un aumento de la tasa de síndrome gripal en mayores de 75 años, de momento no se refleja en las hospitalizaciones. Los datos, además, constatan que continúan en aumento las urgencias pediátricas, sobre todo en niños de entre 5 y 14 años. En estos momentos, la positividad de las muestras centinelas analizadas es del 32,1 por ciento de los casos para el virus de la gripe.
>
> Los patógenos respiratorios más detectados en las muestras centinelas procesadas son los rinovirus/enterovirus y la gripe A (tanto el H1 como el H3). El VRS empieza a circular tanto en las muestras centinelas como en las hospitalarias, con un porcentaje de positividad del 4,8 %. La tasa de incidencia semanal del Covid-19 es de dos por cada 100.000 habitantes, con un 1,9 % de porcentaje de positivos, motivo por el que se encuentra por debajo del umbral epidémico.

Procesamos este texto plano almacenado en `data/raw/vacunacion_cyl.txt` a través de del archivo `pregunta_extra.py` de la misma forma que se procesó la parte obligatoria de esta práctica. Después, los resultados de este procesamiento se guardan en `data/processed/pred_labels_vacunacion.txt` y se comparan los resultados con el texto etiquetado manualmente en `data/gold/gold_vacunacion.txt`.

```
## Carga del modelo
nlp = spacy.load("es_core_news_sm")
nlp.tokenizer=Tokenizer(nlp.vocab)
print("␣Modelo␣es_core_news_sm␣cargado␣correctamente")

## Procesamiento del archivo vacunacion_cyl.txt
```

```
with open("../data/raw/vacunacion_cyl.txt", "r", encoding="utf-8") as file:
    text_vacunacion = file.read()
doc_vacunacion = nlp(text_vacunacion)
print(f"Procesamiento completado: {len(doc_vacunacion)} tokens analizados")

##  Conversi n a formato IOB
spacy_output_vacunacion=text_to_iob(doc_vacunacion)
with open('../data/processed/pred_labels_vacunacion.txt', 'w', encoding='utf-8') as
    out_file:
    out_file.write(spacy_output_vacunacion)
print("Etiquetas predichas guardadas en 'data/processed/pred_labels_vacunacion.txt'")

# Creamos el documento en formato word gold pred
true_labels_file = "../data/gold/gold_vacunacion.txt"
pred_labels_file = "../data/processed/pred_labels_vacunacion.txt"
output_file = "../data/processed/combined_vacunacion.txt"
combine(true_labels_file,pred_labels_file,output_file)

print("Archivo combinado creado: 'data/processed/combined_vacunacion.txt' para realizar
    la evaluaci n")

## Evaluaci n del modelo
with open("../data/processed/combined_vacunacion.txt", "r",encoding="utf-8") as file:
    result = evaluate_conll_file(file)

print("\nRESULTADOS DE LA EVALUACI N:")
print("=" * 50)
print(result)
print("=" * 50)
```

Resultados:

```
accuracy:  57.69%; (non-O)
accuracy:  91.26%; precision:  43.24%; recall:  51.61%; FB1:  47.06
              LOC: precision:  82.35%; recall:  93.33%; FB1:  87.50  17
             MISC: precision:   0.00%; recall:   0.00%; FB1:   0.00  14
              ORG: precision:  20.00%; recall:  20.00%; FB1:  20.00  5
              PER: precision: 100.00%; recall: 100.00%; FB1: 100.00  1

RESULTADOS DE LA EVALUACIÓN:
==================================================
(43.24324324324324, 51.61290322580645, 47.05882352941176)
==================================================
```

La evaluación del modelo de Spacy para este texto muestra un rendimiento limitado y desigual, con un **F1 del 47.06 %** y una **accuracy del 57.69 %**. La **accuracy sobre etiquetas no–O** alcanza el 91.26 %, lo que indica que, cuando el modelo identifica una entidad, suele etiquetarla correctamente.

Análisis por entidad

- **LOC**: Es la categoría con mejor rendimiento, con un F_1 del 87.50 %; el sistema identifica la mayoría de las localizaciones. El texto estaba especialmente indicado para comprobar el rendimiento en esta categoría ya que presenta gran cantidad de localizaciones (con 17) y, pese a eso, el rendimiento es solvente.
- **PER**: Obtiene resultados perfectos (100 %), pero sólo hay un ejemplo, por lo que no es representativo.
- **ORG**: Presenta dificultades claras (F_1 del 20 %).
- **MISC**: El sistema no reconoce ninguna entidad de este tipo (F_1 del 0 %). Este rendimiento es claramente insuficiente.

Los resultados muestran un modelo desigual: funciona bien con categorías como LOC y nos da resultados imprecisos sobre PER. Sin embargo, el rendimiento sobre ORG y MISC es insuficiente.