

**Memoria Trabajo Práctico 1 (TP1)**

Detección de Entidades Nombradas

***Alumno****: Francisco Javier Piqueras Martínez*

***Asignatura****: Minería de Textos*

***Fecha******de******entrega****: 20 de enero de 202.*

Índice

[1. Descripción del documento 3](#_Toc62062457)

[2. Tarea Conll2002 3](#_Toc62062458)

[3. Tarea 4](#_Toc62062459)

[3.1. Entorno 4](#_Toc62062460)

[3.2. Datos 4](#_Toc62062461)

[3.3. Modelo 5](#_Toc62062462)

[3.4. Comparación 6](#_Toc62062463)

[3.5. Evaluación 6](#_Toc62062464)

# Descripción del documento

Este documento consiste en la memoria de la entrega TP1 (Trabajo Práctico 1) de la asignatura de Minería de Textos del Máster en Ingeniería y Ciencia de Datos de la UNED.

Este Trabajo Práctico pertenece los temas 1 y 2 de la asignatura (Introducción al Procesamiento del Lenguaje Natural y Extracción de Información en Documentos).

El enunciado de la tarea es el siguiente:

|  |
| --- |
| *El objetivo de esta práctica es utilizar un etiquetador de entidades nombradas, en español, y evaluar los resultados obtenidos. Se puede utilizar el etiquetador de entidades nombradas de Spacy, que cuenta con un modelo pre-entrenado para el español. Es decir, no se pide entrenar el modelo, solo utilizar las herramientas. El objetivo es ver la utilidad de las herramientas para procesamiento del lenguaje aplicadas a un problema planteado en una competición científica, y analizar los resultados, estudiando las causas de los casos de error y cómo pueden mejorarse.* |

El documento contiene la siguiente información:

* Descripción de la tarea de conll2002 (comentar los principales puntos y resultados del artículo) y los datos de evaluación.
* Descripción del código desarrollado, herramientas utilizadas, etc.
* Textos de prueba utilizados.
* Resultados de evaluación del etiquetado.
* Análisis de los errores de etiquetado y sus causas. Se valorará la introducción de mejoras/modificaciones en el etiquetado que mejoren los resultados.

# Tarea Conll2002

La tarea proporciona un total de 6 archivos (tres para Español y tres para Holandés), los cuales están compuestos por un conjunto de entrenamiento, un conjunto de desarrollo y un conjunto de prueba. Además, están anotados con una variante del esquema IOB categorizando de esta forma cada una de las entidades en: Personas (PER), Organizaciones (ORG), Lugares (LOC) y Miscelánea (MISC).

El objetivo de la tarea [CoNLL 2002](https://www.aclweb.org/anthology/W02-2024/) es plantear un problema de identificación de entidades (NER) multilenguaje y establecer mediante un marco común de evaluación con el que medir las soluciones planteadas.

El marco de evaluación se compone de un baseline y un set de prueba. De esta forma, ejecutaremos las soluciones sobre en conjunto de prueba obteniendo las siguientes métricas:

* Precisión
* Cobertura
* Medida-F

De esta forma, podemos concluir si la solución es real o no. Nos basaremos en el resultado de la solución y la compararemos con el baseline, tomaremos la decisión en función de si se obtiene peor resultado o no.

# Tarea

Como se indica en la tarea, se ha desarrollado un notebook de Jupyter en el que haciendo uso del modelo entrenado previamente de “spaCy”, genera anotaciones IOB sobre el conjunto de prueba. Seguidamente, compara los resultados obtenidos contra las anotaciones del conjunto de prueba.

Finalmente, se obtienen las métricas comentadas anteriormente con el uso del script proporcionado “conlleval.py”.

## Entorno

En primer lugar, se importan todos los módulos en los que se va a trabajar. Se descargan los ficheros que se van a utilizar, ubicándolos en la carpeta “/src”. Se descarga y se inicializa el módulo spaCy.

|  |
| --- |
| # In[1]:  import datetime  import re  import spacy  import spacy.cli  # In[2]:  get\_ipython().system(' wget https://www.clips.uantwerpen.be/conll2002/ner/data/esp.testb -P ./src')  # In[6]:  spacy.cli.download("es")  model = spacy.load('es') |

## Datos

Los datos de entrada presentan la siguiente estructura:

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

Como se puede observar, cada línea contiene una palabra con su correspondiente anotación, siguiente el esquema IOB y con las siguientes anotaciones:

* PER (Personas)
* ORG (Organizaciones)
* LOC (Localizaciones)
* MISC (Miscelánea)

Se abre el conjunto de prueba, ubicado en la carpeta “/src”. Se lee línea por línea.

|  |
| --- |
| # In[3]:  raw\_text = open('./src/esp.testb', 'r')  raw\_text  # In[4]:  full\_text = ''  word\_category = {}  # In[5]:  for i, row in enumerate(raw\_text):  cat\_element = re.split('\s', row)  word\_category[i] = (cat\_element[0], cat\_element[1])  full\_text += cat\_element[0] + ' '  raw\_text.close() |

Cada línea contiene 2 columnas separadas por un espacio, por lo que haremos uso de la expresión regular ‘\s’ para separar cada línea en una palabra y su correspondiente anotación. Asimismo, se guarda todo el texto completo en la variable full\_text.

Además, será necesario tener disponible la información del fichero de referencia junto con sus anotaciones, para ello, se va a almacenar cada palabra y anotación en un diccionario (word\_category) utilizando la posición de la palabra en el fichero como clave.

## Modelo

Para alimentar el modelo sólo se debe pasar la variable “full\_text” como parámetro al modelo inicializado.

|  |
| --- |
| # In[7]:  tokens = model(full\_text) |

El resultado (tokens) contiene el texto anotado con el esquema IOB.

Se almacena en el mismo formato que lo hace SpaCy:

|  |
| --- |
| #In [9]:  token\_output = {}  for i, token in enumerate(tokens):  token\_output[i] = (token.text, token.ent\_iob\_ + "-" + token.ent\_type\_  if token.ent\_iob\_ != 'O'  else token.ent\_iob\_) |

## Comparación

Debemos construir la misma estructura que espera el script “colleval.py”:

“Palabra – Referencia – Anotación.”

|  |
| --- |
| # In[11]:  output\_f = open("./src/output", "w")  # In[14]:  j = 0  for i in word\_category:  word\_ = word\_category[i][0]  cat\_ = word\_category[i][1]    if(word\_.strip() != token\_output[j][0].strip()):  j+=1;  print('The tokens ' + word\_.strip() + ' and ' + token\_output[j][0].strip() + ' do not match.')  else:  res\_ = '' if word\_.strip() == '' else token\_output[i][1]  output\_f.write('{} {} {}\n'.format(word\_, cat\_, res\_))    j+=1;    output\_f.close() |

## Evaluación

Pasamos el contenido por el script “colleval.py” para su evaluación:

|  |
| --- |
| # In[16]:  get\_ipython().system(' python src/conlleval.py < src/output')  # Out[16]  processed 52821 tokens with 3535 phrases; found: 3671 phrases; correct: 76.  accuracy: 2.82%; (non-O)  accuracy: 78.55%; precision: 2.07%; recall: 2.15%; FB1: 2.11  LOC: precision: 1.91%; recall: 2.52%; FB1: 2.17 1413  MISC: precision: 1.07%; recall: 1.80%; FB1: 1.34 560  ORG: precision: 3.17%; recall: 1.94%; FB1: 2.40 853  PER: precision: 1.89%; recall: 2.18%; FB1: 2.03 845 |

Los resultados no son buenos. El modelo ha detectado muy bien las palabras que no pertenecen a entidades (“O”). No obstante, no ha sido capaz de clasificar las palabras.