

# Named Entity Recognition (NER consola UNIX) manual de usuario.

## Aplicación del software

El software de reconocimiento de entidades nombradas (NER), es un programa tipo bash que implementa técnicas de inteligencia artificial para extraer información relevante de un documento de forma automática. Es por esto por lo que usa tecnologías de última generación para desarrollar un framework que permita utilizarlas y presentarlas en un servicio fácil de usar.

El software permite encontrar entidades dentro de un documento de forma automática, de este modo agiliza y automatiza la tarea de identificar Personas, Lugares, Documentos, etc. Toda la información de las entidades encontradas en un documento se guardará en otro documento que podrá ser usado posteriormente para el análisis o procesamiento de información relevante.

El software cuenta con un modelo por defecto que está definido dentro del contexto comercial de la cámara de comercio de Colombia. De igual manera permite realizar el entrenamiento de nuevos modelos para otros contextos de información diferente, así se genera el modelo para su posterior uso.

Así el software agiliza la implementación de técnicas de procesamiento de lenguaje natural para el análisis de documentos por medio de comandos de consola.

## Índice.

Sección	Página
<b>1. Como instalar</b>	<b>2</b>
<b>2. Descripción del Software</b>	<b>2</b>
<b>3. Cómo usarlo (Ejemplos de comandos)</b>	<b>3</b>
<b>4. Formatos</b>	<b>5</b>
<b>4.1. Documento de entrada</b>	<b>5</b>
<b>4.2. Documento de salida</b>	<b>5</b>
<b>4.3. CONLL-03</b>	<b>8</b>
<b>4.4. PRATECH</b>	<b>8</b>
<b>5. Solución de errores</b>	<b>11</b>
<b>6. Glosario</b>	<b>11</b>

## Cómo instalar

Para su instalación es necesario contar con Docker dentro del servidor o computador en el cual se instalará.

Desde el directorio principal donde se encuentra el Dockerfile, se debe ejecutar las siguientes líneas de comando en la consola del sistema:

- `docker build -t ner .`
- `docker run -it --name ner_container --mount type=bind,source="$(pwd)"/data,target=/workspace/data ner`

luego ejecutar el comando deseado en la consola.

## Descripción del software

El software se controla por un comando bash. Este comando tiene 2 modos de uso: (1) modo *TRAIN* para entrenar un nuevo modelo y (2) modo *USE* para usar un modelo existente.

Para el modo *TRAIN* podemos controlar los siguientes parámetros:

Parámetro	Descripción
-f	Entrenamiento rápido (solo para motivos de prueba, el modelo resultante no será útil). <i>True</i> para activar, <i>False</i> para desactivar. <b>Parámetro opcional, por defecto False.</b>
-m	Nombre del nuevo modelo. <b>Nota: Si se usa el nombre de un modelo existente este se sobrescribirá</b> <b>Parámetro obligatorio.</b>
-s	Tipo de formato de los documentos de entrada en, se puede seleccionar formato estándar (CONLL-03) con <i>True</i> o PRATECH con <i>False</i> . <b>Nota: Formatos definidos en la sección Formatos</b> <b>Parámetro opcional, por defecto False.</b>
-id	Directorio de datos de entrenamiento, debe ser una carpeta con solo los archivos de entrenamiento, ya sea formato CONLL-03 o PRATECH. <b>Parámetro obligatorio.</b>
-u	Sobre muestreo. Aplica técnicas de sobre muestreo a los datos de entrada. <i>True</i> para activar, <i>False</i> para desactivar. <b>Parámetro opcional, por defecto False.</b>
-cu	Analiza si hay una GPU disponible, si es así la usará, si no usará la CPU. <i>True</i> para activar y buscar GPU, <i>False</i> para usar CPU. <b>Parámetro opcional, por defecto False.</b>

En la **Figura 1** se muestra un ejemplo del comando en modo *TRAIN*:

```
./Bash_handler.sh TRAIN -m Nuevo_modelo -id "/mnt/c/Users/sanmo/OneDrive - Universidad de Antioquia/GITA/Maestría/Programas/Software NER/data/train/fast"
```

**Figura 1.** Modo *TRAIN*

Para el modo *USE* podemos controlar los siguientes parámetros:

Parámetro	Descripción
-m	Nombre del modelo a usar. (Por defecto el programa cuenta con el modelo CCC) <b>Parámetro obligatorio.</b>
-id	Ruta del documento al cual se le aplicará el NER, debe ser un archivo. <b>Nota:</b> Debe estar en el formato de entrada (Definido en la sección Formatos). <b>Parámetro obligatorio.</b>
-od	Ruta del documento donde se guardarán los resultados, debe ser un archivo. <b>Nota:</b> Estará en formato de salida (Definido en la sección Formatos). <b>Parámetro opcional por defecto data/tagged/document_tagged.json.</b>
-cu	Analiza si hay una GPU disponible, si es así la usará, si no usará la CPU. True para activar y buscar GPU, False para usar CPU. <b>Parámetro opcional, por defecto False.</b>

En la **Figura 2** se muestra un ejemplo del comando en modo *USE*:

```
./Bash_handler.sh USE -m CCC -id "/mnt/c/Users/sanmo/OneDrive - Universidad de Antioquia/GITA/Maestría/Programas/Software NER/prueba.json"
```

*Figura 2. Modo USE*

## Cómo usarlo (Ejemplos de comandos)

### Reconocimiento de entidades en un documento usando el modelo CCC

```
./Bash_handler.sh USE -m CCC -id "/mnt/c/Users/sanmo/OneDrive - Universidad de Antioquia/GITA/Maestría/Programas/Software NER/prueba.json"
```

### Reconocimiento de entidades en un documento usando el modelo CCC, con un archivo de salida específico.

```
./Bash_handler.sh USE -m CCC -id "/mnt/c/Users/sanmo/OneDrive - Universidad de Antioquia/GITA/Maestría/Programas/Software NER/prueba.json" -od "/mnt/c/Users/sanmo/OneDrive - Universidad de Antioquia/GITA/Maestría/Programas/Software NER/data/prueba_salida.json"
```

### Reconocimiento de entidades en un documento usando el modelo CCC, con un archivo de salida específico y usando GPU.

```
./Bash_handler.sh USE -m CCC -id "/mnt/c/Users/sanmo/OneDrive - Universidad de Antioquia/GITA/Maestría/Programas/Software NER/prueba.json" -od
```

```
"/mnt/c/Users/sanmo/OneDrive - Universidad de Antioquia/GITA/Maestría/Programas/Software  
NER/data/prueba_salida.json" -cu True
```

**Entrenamiento de un nuevo modelo llamado *New\_model* con documentos formato no estándar (PRATECH).**

```
./Bash_handler.sh TRAIN -m Nuevo_modelo -id "/mnt/c/Users/sanmo/OneDrive - Universidad de  
Antioquia/GITA/Maestría/Programas/Software NER/data/train/fast"
```

**Entrenamiento rápido de un nuevo modelo llamado *New\_model* con documentos formato estándar (CONLL-03), haciendo uso de GPU y con sobre muestreo.**

```
./Bash_handler.sh TRAIN -f True -m Nuevo_modelo -s True -id "/mnt/c/Users/sanmo/OneDrive –  
Universidad de Antioquia/GITA/Maestría/Programas/Software NER/data/train" -u True -cu True
```

**Nota:** si se desea obtener el archivo del nuevo modelo se debe mover el archivo a la carpeta *data*.

## Formatos.

Para el modo *USE*:

### Documento de entrada.

El formato de entrada debe contener un objeto cuyo nombre “sentences” y su valor es una lista de bloque donde cada bloque tiene un objeto que lleva por nombre “text” y su valor es el texto de la frase.

En la **Figura 3** se muestra un ejemplo del formato de entrada

```
{
  sentences: [
    0: {
      text: "Frase ejemplo"
    },
    1: {
      text: "Frase ejemplo"
    }
  ]
}
```

**Figura 3.** Formato documento de entrada

### Documento de salida.

El documento de salida cuenta con 4 objetos de nombres: text, text\_labeled, sentences y entities.

- El objeto **text**, tiene por valor el texto plano del documento.
- El objeto **text\_labeled** tiene por valor el texto plano etiquetado de la siguiente manera. Si un token del texto tiene una etiqueta diferente a “O”, el token se encierra en comillas y se le adiciona un ‘/’ seguido de la etiqueta correspondiente.
- El objeto **sentences** tiene por valor una lista de bloques, cada bloque de la lista hace referencia a una frase del documento y está compuesto de los siguientes 3 objetos:
  - **text**: tiene por valor el texto plano de la frase
  - **text\_labeled**: tiene por valor el texto plano etiquetado de la misma forma que el objeto text\_labeled del documento.
  - **Tokens**: tiene por valor una lista de bloques, cada bloque de la lista hace referencia a un token de la frase y está compuesto por: (1) un objeto de nombre “text” que contiene por valor el texto plano del token y (2) un objeto de nombre label que contiene por valor la etiqueta de dicho token.
- El objeto **entities** tiene por valor una lista de bloques, donde cada bloque se refiere a cada mención de cada entidad encontrada en el documento, está compuesto de los siguientes objetos:

- **entity:** tiene por valor el nombre de la entidad.
- **index:** tiene por valor la posición a nivel de token dentro del texto plano del token al cuál la entidad se refiere.
- **word:** tiene por valor el texto del token.
- **start:** tiene por valor la posición a nivel de carácter del primer carácter del token
- **end:** tiene por valor la posición a nivel de carácter del último carácter del token

En la **Figura 4** se encuentra un ejemplo del formato de salida.

```

{
  text: "Frase ejemplo Frase ejemplo ",
  text_labeled: " "Frase"/Entity_Type ejemplo "Frase"/Entity_Type ejemplo ",
  sentences: [
    0: {
      text: "Frase ejemplo",
      text_labeled: " "Frase"/Entity_Type ejemplo",
      tokens: [
        0: {
          text: "Frase",
          label: "Entity_Type"
        },
        1: {
          text: "ejemplo",
          label: "0"
        }
      ]
    }
  ],
  1: {
    text: "Frase ejemplo",
    text_labeled: " "Frase"/Entity_Type ejemplo",
    tokens: [
      0: {
        text: "Frase",
        label: "Entity_Type"
      },
      1: {
        text: "ejemplo",
        label: "0"
      }
    ]
  }
],
  entities: [
    0: {
      entity: "Entity_Type",
      index: 0,
      word: "Frase",
      start: 0,
      end: 5
    },
    1: {
      entity: "Entity_Type",
      index: 2,
      word: "Frase",
      start: 14,
      end: 19
    }
  ]
}

```

**Figura 4.** Formato documento de salida

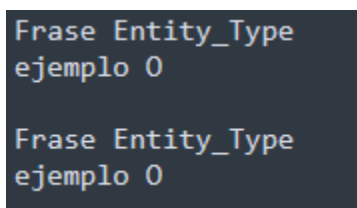
Para el modo *TRAIN*:

### CONLL-03

El formato CONLL-03 consiste en un documento de texto donde se encuentran diversas frases. Cada frase está compuesta por diversos tokens. Para el formato cada línea del documento corresponde a un token y su respectiva entidad separados por un espacio. La separación entre frases está dada por un salto de línea.

**Nota:** Para el uso de este formato el directorio de entrada debe contener 2 archivos: (1) *Train.txt*, el cual contendrá los datos de entrenamiento y (2) *Test.txt* el cual contendrá los datos de prueba. Ambos archivos deben estar en formato CONLL-03.

En la **Figura 5** se muestra un ejemplo de un archivo en formato CONLL-03.



```
Fraser Entity_Type
ejemplo 0

Fraser Entity_Type
ejemplo 0
```

**Figura 5.** Formato CONLL-03

### PRATECH

El formato PRACTECH cuenta con 3 objetos de nombres: text, sentences y mentions.

- El objeto **text**, tiene por valor el texto plano del documento.
- El objeto **sentences** tiene por valor una lista de bloques, cada bloque de la lista hace referencia a una frase del documento y está compuesto de los siguientes 3 objetos:
  - **text**: tiene por valor el texto plano de la frase
  - **id**: tiene por valor un identificador de la posición de la frase dentro del documento, de la forma *s#*.
  - **Tokens**: tiene por valor una lista de bloques, cada bloque de la lista hace referencia a un token de la frase y está compuesto por: (1) un objeto de nombre "text" que contiene por valor el texto plano del token, (2) un objeto de nombre begin que contiene por valor la posición a nivel de carácter del primer carácter del token y (3) un objeto de nombre end que contiene por valor la posición a nivel de carácter del último carácter del token.
- El objeto **mentions** tiene por valor una lista de bloques, donde cada bloque se refiere a cada mención de cada entidad encontrada en el documento, está compuesto de los siguientes objetos:



- **id:** tiene por valor un identificador del número de frase y el número de mención a la que hace referencia., de la forma s#-m#.
- **type:** tiene por valor el nombre de la entidad.
- **word:** tiene por valor el texto del token.
- **begin:** tiene por valor la posición a nivel de carácter del primer carácter del token
- **end:** tiene por valor la posición a nivel de carácter del último carácter del token

**Nota:** Para el uso de este formato el directorio de entrada debe contener solo documentos JSON en el formato PRATECH y debe contener al menos 2.

En la **Figura 6** se encuentra un ejemplo del formato de salida.

```

{
  text: "Frase ejemplo Frase ejemplo",
  sentences: [
    0: {
      text: "Frase ejemplo",
      id: "s0",
      tokens: [
        0: {
          text: "Frase",
          begin: 0,
          end: 5
        },
        1: {
          text: "ejemplo",
          begin: 6,
          end: 13
        }
      ]
    },
    1: {
      text: "Frase ejemplo",
      id: "s1",
      tokens: [
        0: {
          text: "Frase",
          begin: 14,
          end: 19
        },
        1: {
          text: "ejemplo",
          begin: 20,
          end: 27
        }
      ]
    }
  ],
  mentions: [
    0: {
      id: "s0-m0",
      type: "Entity_type",
      begin: 0,
      end: 5
    },
    1: {
      id: "s1-m0",
      type: "Entity_type",
      begin: 14,
      end: 19
    }
  ]
}

```

**Figura 6.** Formato PRATECH

## Solución de errores.

En la siguiente tabla se encuentra la información de los posibles errores, con su código de identificación, su causa y su posible solución

Código	Causa	Solución
1	Error cargando el modelo	Reentrenar el modelo verificando su completa finalización.
2	Error cargando el documento de entrada para predicción.	Revisar el documento de entrada ya que este puede estar vacío o corrupto.
3	Error en el formato del documento de entrada para predicción.	Revisar el formato del documento de entrada ya que no corresponde al formato que acepta el software.
4	Carpeta de documentos para entrenamiento vacía.	Usar una carpeta en donde se encuentren los documentos para el entrenamiento.
5	Error cargando el modelo base desde la nube para el entrenamiento.	Revisar la conexión a internet y reiniciar el entrenamiento.
7	Error en el entrenamiento del modelo.	Revisar la conexión a internet y reiniciar el entrenamiento.
8	Error en los archivos de entrada para el entrenamiento.	Revisar el formato de cada uno de los documentos de entrada para el entrenamiento.
9	La ruta de entrada para predicción de documento no corresponde a un archivo.	Verificar que la ruta existe y finalice en un archivo JSON.
10	El modelo seleccionado no existe.	Seleccionar un modelo existente.
11	Error en el documento de salida para predicción.	Verificar que la ruta existe y finalice en un archivo JSON.

**Tabla 1.** Resumen de errores.

## Glosario.

**NER:** Reconocimiento de entidades nombradas. NER por sus siglas en inglés (Named Entity Recognition).

**CPU:** Unidad central de procesamiento. CPU por sus siglas en inglés (Central Processing Unit).

**GPU:** Unidad de procesamiento gráfico. GPU por sus siglas en inglés (Graphics Processing Unit).

**JSON:** Notación de objeto de JavaScript es un formato de texto sencillo para el intercambio de datos. JSON por sus siglas en inglés (JavaScript Object Notation).

**Framework:** es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.