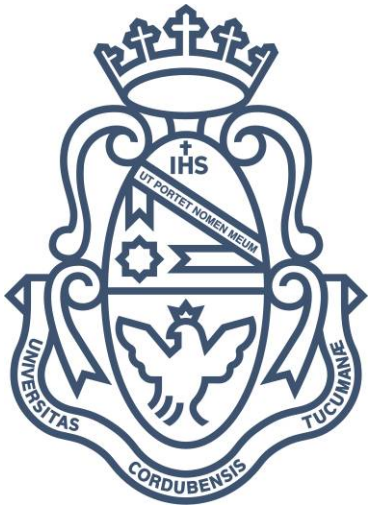
Facultad de Matemática, Astronomía, Física y Computación

Universidad Nacional de Córdoba



Mejorando reconocimiento de entidades nombradas del Español mediante la especialización de BETO

Tesis para obtener el título de

Licenciado en Ciencias de la Computación

Autor: Nazareno Garagiola

Director: Cristian Cardellino

Córdoba, Argentina 2022

Agradecimientos

A mi familia, amigos, compañeros y docentes ¡Muchas Gracias!

Resumen

En este trabajo se realizarán tareas de especialización sobre *BETO*, que es un modelo de lenguaje no supervisado, equivalente al modelo del lenguaje *BERT*, pero entrenado sobre un corpus de gran volumen del Español. El entrenamiento de BETO es no supervisado, y está pensado para servir como base a diferentes tareas mediante un proceso de afinado sobre dichas tareas. En este trabajo en particular, intentaremos mejorar los resultados de la tarea de Reconocimiento de Entidades Nombradas del Español. BETO ha demostrado tener un buen desempeño utilizando como corpus de evaluación a *CONLL*.

Este trabajo contará de varias etapas, cuyo objetivo final será mejorar los resultados de base establecidos por BETO en reconocimiento de entidades nombradas para el Español. En una primera etapa se buscará reproducir los resultados de BETO para tener un resultado sobre el cuál desarrollar. En una siguiente etapa se buscará mejorar el desempeño de BETO utilizando grandes corpus anotados del Español, que requerirán un pre-proceso para alinearlo a las entidades presentes en el corpus de evaluación. En una última etapa, se pasará a un esquema semi-supervisado, donde se utilizarán los mejores modelos encontrados para anotar un conjunto de datos no etiquetado, que se utilizará para afinar BETO en un esquema de bootstrapping.

Índice general

Capítulo 1

Introducción y motivación

1.1. Introducción

La extracción de información es una tarea fundamental en el análisis de texto libre para poder estructurar dichos datos de manera automática y poder así democratizar el acceso a dicha información. Una de las tareas más importantes que se engloban dentro de la gran área que es la extracción de información, es el reconocimiento de entidades nombradas.

El *Reconocimiento de Entidades Nombradas* (NER, por sus siglas en inglés) es la tarea de localizar y categorizar sustantivos propios (nombres de personas, lugares, empresas, etc) en un documento. Como tantas otras áreas del Procesamiento de Lenguaje Natural, existen buenos y variados recursos para realizar una tarea específica de forma automática en idioma Inglés. No obstante, muchas veces al cambiar el idioma, los recursos (e.g. cantidad o calidad de datos anotados) y el desempeño de los modelos suele verse afectado.

Parte de la dificultad de la tarea NER es que, a diferencia de otras tareas de clasificación (como el etiquetado gramatical, o *Part-of-Speech Tagging* en inglés), cada palabra a identificar puede ser parte o no de una entidad nombrada, una misma palabra puede hablar de un nombre de una empresa o de una persona (por ejemplo, Phillip Morris). También ocurren casos en los que una palabra puede confundirse entre ser parte de un nombre o solo ser una palabra que habla sobre el contexto (por ejemplo, Vicente del Bosque).

En los últimos años, modelos pre-entrenados, basados en la arquitectura del *Transformer* (Vaswani et al., 2017) han tenido gran impacto en muchas de las tareas de procesamiento de lenguaje natural. En particular, los modelos bidireccionales, como *BERT* (Devlin et al., 2018), han demostrado buenos resultados en tareas de clasificación de texto como el reconocimiento de entidades nombradas.

La librería *Hugging Face Transformers* (Wolf et al., 2020) ha hecho posible que dichos modelos sean accesibles a quienes estén interesados en experimentar con ellos y buscar mejorar los resultados. Como consecuencia, se han empezado a generar muchos modelos especializados para tareas específicas, como es el caso de reconocimiento de entidades nombradas. Aún así, no existen muchos modelos especializados en reconocimiento de entidades nombradas basados en *BERT* para el idioma Español.

El objetivo de este trabajo es la exploración de métodos que ayuden a mejorar los resultados de la tarea de reconocimiento de entidades nombradas para el idioma Español.

La manera de lograrlo será mediante la *afinación* (o *fine-tuning* en inglés) de un modelo de lenguaje basado en BERT, pero pre-entrenado sobre un corpus del Español. Dicho modelo es BETO (Cañete et al., 2020), que se encuentra disponible en el repositorio de Hugging Face, pero no está especializado para el reconocimiento de entidades nombradas. El modo de trabajo será, en primera instancia, la replicación de los resultados obtenidos en el trabajo de (Cañete et al., 2020) para el reconocimiento de entidades, evaluando sobre el corpus del español CONLL (Tjong Kim Sang, 2002). En segunda instancia, se buscará experimentar con ejemplos de los conjuntos WikiNER (Nothman et al., 2017) y WikiNEuRal (Tedeschi et al., 2021), que tienen una cantidad mucho mayor de ejemplos anotados automáticamente, con el objetivo de ver cómo impacta en el desempeño del modelo evaluado en CONLL. Finalmente, con el objetivo principal de expandir el conjunto de entrenamiento, para de esa manera tener una mayor cobertura de casos, se buscará la combinación entrenando con ejemplos tanto de CONLL como de WikiNER/WikiNEuRal, en un esquema semi-supervisado de bootstrapping. El mejor modelo obtenido se subirá al repositorio de Hugging Face, para que esté disponible al público, un modelo basado en BETO y afinado para reconocimiento de entidades nombradas del Español.

1.2. Motivación

El problema que trae consigo el reconocimiento de entidades nombradas ya tiene varios años, pero está lejos de ser solucionado especialmente para idiomas fuera del Inglés. En el articulo (Sundheim, 1995) que fue parte de una competencia sobre la tarea de NER, se destaca que nombres comunes de organizaciones, primeros nombres y nombres de lugares pueden ser manejados recurriendo a listas de nombres, empresas o a un Nomenclátor [1] aunque existen inconvenientes. Entre estos inconvenientes se encuentran: la poca disponibilidad de estos grandes conjuntos de datos, el determinar cuál es el menor conjunto de nombres necesarios, la deficiencia que se observa al cambiar de dominios (i.e. usar el mismo conjunto de datos para un texto diferente) y la escasez de estos recursos en otros idiomas fuera del inglés.

Por ejemplo, de acuerdo al listado de nombres posibles en la Provincia de Córdoba existen más de 11600 primeros nombres posibles, considerando las posibles combinaciones de nombres únicos con apellidos, tendríamos un lista enorme de nombres solo para identificar nombres de personas, sumado al reconocimiento de empresas o lugares, se vuelve una forma de trabajar ineficiente.

Actualmente existen diversas formas de lidiar con la tarea NER de forma automática, tales como se describen en distintos trabajos (Morwal et al., 2012), (Li et al., 2008), pero en los últimos años ha tomado mucha fuerza el uso de modelos basados en *Transformers*. De estos, se destacan los modelos bidireccionales como BERT (Devlin et al., 2018); los cuales poseen la ventaja de poder clasificar una entidad gracias a la capacidad para "ver a ambos lados" de cada palabra, la cual fue explotada utilizando *BiLSTM* (Lample et al., 2016).

BERT es un modelo pre-entrenado en una tarea de modelado de lenguaje, en este caso particular es el modelado enmascarado (o *masked language model* en inglés) dónde se trata de predecir palabras que son ocultadas al azar, a partir del contexto. Esta tarea

[1] Índice geográfico o directorio

de pre-entrenamiento luego se puede utilizar como base para "afinar" el modelo con una tarea como objetivo específica (e.g. reconocimiento de entidades, etiquetado gramatical, clasificación de oraciones, etc.), lo que ahorra la necesidad de un entrenamiento desde cero, algo complejo fuera del idioma Inglés, donde hay escasez de recursos anotados. Para el Español tenemos *BETO* (Cañete et al., 2020), que es un modelo basado en *BERT*, pero este está pensado para múltiples tareas de Procesamiento de Lenguaje Natural. Es por eso que la idea de este trabajo es explorar el modelo pre-entrenado y mejorar los resultados de la tarea de reconocimiento de entidades nombradas del Español.

Si bien existen trabajos que intentan combinar ambas opciones, mediante un procesamiento de listas de entidades para poder utilizarlos como una característica más entrenar una red neuronal (Magnolini et al., 2019), en nuestro trabajo nos enfocaremos en continuar explorando las posibilidades de la arquitectura de *BETO* bajo distintos corpus anotados del Español, primero realizando un aprendizaje automático supervisado y luego adoptando un enfoque de aprendizaje semi-supervisado.

1.3. Esquema de la tesis

Este trabajo de tesis se estructura de la siguiente manera: En el Capítulo 2 se introducen trabajos previos relacionados que sirven de base para este trabajo, las métricas a utilizar, los conjuntos de datos sobre los cuales entrenaremos los modelos y una introducción a *Hugging Face*. En el Capítulo 3 se explica el trabajo realizado y las decisiones que diferencian este trabajo de los demás mencionados en el Capítulo 2. Luego, en el Capítulo 4 describimos los experimentos realizados y analizamos los resultados obtenidos en las pruebas, haremos una comparación analítica de las métricas y valores obtenidos, junto a un análisis cualitativo a partir de algunos textos en específico. Finalmente en el Capítulo 5 presentamos las conclusiones finales sobre el proyecto y los resultados, además de establecer algunas posibles líneas de investigación para trabajos futuros.

Capítulo 2

Trabajo Relacionado

En este capítulo se presentan las herramientas y trabajos previos que servirán como base para nuestro propio trabajo. Primero introducimos los trabajos que nos permitieron el desarrollo del modelo que luego refinaremos en el entrenamiento, los conjuntos de datos que conseguimos y luego las funcionalidades de Hugging Face que usaremos para nuestra investigación.

2.1. Trabajos previos

La tarea de Reconocimiento de Entidades Nombradas (NER), como aclaramos antes, es la tarea de encontrar palabras que reflejen el nombre de alguna entidad, ya sean personas, empresas, lugares, etc. Dentro del área de extracción de información, la tarea de reconocimiento de entidades es una tarea de clasificación de secuencias, la cual se diferencia de las tareas de Clasificación de Texto ya que en esta última se busca clasificar un texto completo como un documento, en vez de clasificar cada palabra del documento.

2.1.1. Tarea Conjunta CONLL-2002

Desde sus inicios han existido competencias para alentar el desarrollo del área de reconocimiento de entidades nombradas, tal es el caso de la tarea conjunta propuesta por *CONLL*[1] en 2002(Tjong Kim Sang, 2002).

Para esta competencia se generaron conjuntos de datos en los idiomas Holandés y Español, siendo el segundo el único dataset en Español anotado manualmente disponible en la actualidad. Los datos contienen entidades que se separan en las clases:

- PER: clase para nombres de personas.
- ORG: son nombres de Organizaciones, empresas, entidades bancarias, etc.
- LOC: nombres de ciudades, provincias, países, etc.
- Misceláneo: nombres propios que no entran dentro de las clases anteriores.

[1] `https://conll.org/`

Estos datos fueron anotados bajo el formato *IOB* (Ramshaw and Marcus, 1995), que define la regla: *B-etiqueta* si la palabra esta al comienzo de una entidad, *I-etiqueta* si es una palabra dentro de una entidad nombrada pero no es el primero u *O* en caso contrario.

Por ejemplo, la frase:

```
Ignacio hizo un viaje por la ciudad de Buenos Aires.
```

contiene las entidades "Ignacio" y "Buenos Aires", la primera una persona y la segunda un lugar. Al etiquetarse, quedaría de la siguiente forma:

```
Ignacio[B-PER] hizo[O] un[O] viaje[O] por[O] la[O] ciudad[O] de[O] Buenos[B-
LOC] Aires[I-LOC] .[O]
```

Además de la creación de los datos, para la competencia se definieron las métricas que aún hoy se siguen utilizando para evaluar los modelos de NER, estas son:

- *Precision* es el número de entidades nombradas que coinciden exactamente con el conjunto de evaluación.
- *Recall (o exhaustividad)* es el número de entidades del conjunto de evaluación que aparecen exactamente en la misma posición en las predicciones.
- *F1-Score* está definida como la media armónica de estos dos valores y sera la métrica usada para evaluar nuestros experimentos.

2.1.2. Corpus anotados automáticamente

Parte de nuestro trabajo fue también el de encontrar nuevos conjuntos de datos anotados en Español para poder utilizar en el entrenamiento de nuestros modelos. Presentamos entonces los conjuntos de datos de WikiNER (Nothman et al., 2017) y de WikiNEuRal (Tedeschi et al., 2021). Estos son datos recopilados de la Wikipedia, los cuales fueron etiquetados mediante diferentes técnicas de forma automática.

El primero utilizando un enfoque top-down de etiquetado, primero clasificando artículos completos y luego transformando los hipervínculos entre las anotaciones a etiquetas proyectando la clase del articulo. Esto mejora la velocidad para el proceso de etiquetado, pero tiene dos contrapuntos: no todos los textos de la Wikipedia contienen todos los hipervínculos correctamente anotados, y el mapeo de las etiquetas de artículos a las clases utilizadas no es trivial.

Mientras que WikiNEuRal funciona utilizando *Multilingual BERT* dentro de su esquema de trabajo para poder identificar entidades, añade una capa de BiLSTM (bi-diretional Long Short Term Memory) junto a un modelo de CRF (Conditional Random Fields) y utiliza el trabajo de WikiNER para aumentar la cantidad de anotaciones.

El conjunto de datos de WikiNEuRal se adecúa correctamente al formato que mostramos en la sub-sección 2.1.1, mientras que para el uso de los datos de WikiNER fue necesario obtener los archivos originales desde la web [2], hacer un pre-procesamiento para

[2] `https://figshare.com/articles/dataset/Learning_multilingual_named_entity_recognition_from_Wikipedia/5462500`

actualizar su formato y transformar las etiquetas del formato *IO* (i.e. que no distinguen la primera palabra de la entidad nombrada del resto) al *IOB* que utilizamos en los demás conjuntos de datos.

Además, en el año 2020, dentro del marco de una competencia de *IberLEF*,[3] se realizó una nueva investigación para fomentar el desarrollo del procesamiento de lenguaje natural en el idioma Español junto a otros idiomas ibéricos. Para este se anoto parte de un Corpus de CAPITEL para ser usado en una competencia sobre reconocimiento de entidades nombradas (Porta and Espinosa-Anke, 2020). Lamentablemente este conjunto de datos no está disponible para su uso público, por lo que no fue posible conseguirlo para el desarrollo de este trabajo.

2.1.3. BETO

Desde el año de la competencia de CONLL, han surgido nuevos desarrollos que han cambiado la forma de trabajar dentro del área de Extracción de Información (Finkel et al., 2005) y técnicas específicas para la tarea de reconocimiento de entidades basados en modelos bidireccionales (Lample et al., 2016), los cuales permiten que la predicción de una palabra dependa de las demás palabras en la secuencia.

Desde el desarrollo de *BERT* (Devlin et al., 2018), una arquitectura basada en una pila de Transformers Encoders que utilizan Self-Attention(Vaswani et al., 2017), se comenzaron a utilizar modelos con este estilo, que se pre-entrenan con un gran conjunto de datos. Para poder permitir utilizar el contexto en ambas direcciones, en el pre-entrenamiento se utiliza un Modelo de Lenguaje Enmascarado (Masked Language Model), este consta de reemplazar por un token especial a un 15 % de los tokens de una secuencia. Por ejemplo, si entrenamos con la frase que utilizamos previamente:

```
Ignacio hizo un viaje por la ciudad de Buenos Aires.
```

podría resultar en:

```
Ignacio hizo un [MASK] por la ciudad de Buenos [MASK].
```

Luego este modelo puede refinarse con un conjunto de datos en específico para una tarea particular.

En nuestro caso para el idioma Español, dos modelos basados en BERT se encuentran disponibles actualmente, uno es *BETO* (Cañete et al., 2020) y el otro es *M-BERT*[4]. El segundo, desarrollado por *Google* en conjunto con el modelo original de BERT, si bien presenta la posibilidad de diferentes lenguajes, contiene deficiencias al moverse de un idioma al otro (Pires et al., 2019). Por esta razón, utilizamos BETO para nuestro trabajo, el cual fue puramente entrenado con un corpus en Español.

BETO fue presentado como el primer modelo basado en BERT pre-entrenado con casi 3 mil millones de palabras en español de diferentes fuentes (Wikipedia, subtítulos, charlas TED, etc) y luego fue testeado en diferentes tareas de procesamiento de lenguaje natural bajo el espíritu de la competencia GLUE (Wang et al., 2018). Actualmente se presentaron dos modelos: BETO Cased, el cual contiene palabras con mayúsculas y BETO Uncased, donde todas las palabras se pasaron a minúsculas para el pre-entrenamiento. Cada modelo contiene un vocabulario de 32000 tokens o sub-palabras.

[3] `https://sites.google.com/view/capitel2020`

[4] `https://github.com/google-research/bert/blob/master/multilingual.md`

2.1.4. Comparación de resultados sobre CONLL

Comparamos los mejores resultados de la métrica F1-Score obtenidos en los trabajos de WikiNER, WikiNEuRal y BETO al evaluarse sobre el conjunto de datos de evaluación de la competencia CONLL-2002. Estos se pueden ver en el cuadro 2.1.

Nombre de modelo	F1-Score
BETO Uncased	82.67
BETO Cased	**88.43**
WikiNER	87.70
WikiNEuRal	86.49

Cuadro 2.1: Resultados del conjunto de test de CONLL-2002 obtenido por los trabajos previos.

2.2. Hugging Face

La compañía *Hugging Face* comenzó como un chat para adolescentes que utilizaba desarrollos en el área de procesamiento de lenguaje natural con código abierto (open source), pero luego fue evolucionando hasta volverse una librería con el objetivo de democratizar el uso e investigación del procesamiento del lenguaje.

Tras el desarrollo de *BERT*(Devlin et al., 2018) por *Google*, Hugging Face comenzó a ofrecer modelos basados en esta arquitectura para las diferentes tareas del área, como clasificación de texto, Extracción de Información y demás. Actualmente tienen a disposición un sitio web[5] donde es posible registrarse para utilizar las librerías que proveen.

En el marco de este trabajo nos interesa destacar dos funcionalidades claves que la plataforma permite. La primera es la posibilidad de subir a un repositorio el conjunto de datos que se utilice para entrenar un modelo, permitiendo uniformidad en los datos. La otra característica que utilizamos es la API de *Trainer*[6], con la cual podemos entrenar cualquier modelo disponible de manera sencilla, eficiente y con diversas funcionalidades: permite elegir hiper-parámetros, agregar funciones personalizadas durante el ciclo de entrenamiento, uso de CPU/GPU/TPU según las disponibilidades de hardware, seleccionar diferentes conjuntos de datos para el entrenamiento, validación o evaluación, distintos optimizadores, entre otras cosas.

[5] `https://huggingface.co/`

[6] `https://huggingface.co/docs/transformers/main_classes/trainer`

Capítulo 3

Metodología de Trabajo

En este capítulo detallamos las pautas que guiaron nuestro trabajo, los conjuntos de datos utilizados y el pre-proceso necesario para su uso. También se presentan las decisiones de diseño tomadas para el entrenamiento del modelo junto a las dificultades encontradas en el camino y cómo se solucionaron.

3.1. Esquema de Trabajo

3.1.1. Obtener modelo de BETO

En primer lugar, utilizando la API *Transformers*[1] de *Hugging Face* se obtuvo el modelo base de BETO junto con su Tokenizador, que nos ayuda a procesar las palabras.

El *Tokenizador* es una abstracción que nos permite pasar de palabras a *tokens*. Si bien en la literatura clásica de procesamiento de lenguaje natural, los tokens suelen ser considerados como las unidades mínimas que conforman un vocabulario, en el contexto de BERT (y consecuentemente BETO), estos son partes de palabras (o sub-palabras) que son reconocidos por el modelo en cuestión. Esto se realiza mediante un método de tokenización de sub-palabras, en particular BERT (y BETO) utilizan el algoritmo de WordPiece Tokenization (Schuster and Nakajima, 2012), que se conforma de un modelo finito de sub-palabras (o tokens) que se utilizan para representar la totalidad de palabras del corpus de entrenamiento y se puede utilizar para representar también palabras fuera de vocabulario. La idea de utilizar algoritmos de tokenización de sub-palabras es evitar tener que lidiar con un vocabulario muy amplio, ahorrando de esa manera tiempo y memoria de cómputo (además de evitar problemas con palabras fuera de vocabulario).

Para demostrar cómo funciona el tokenizador podemos tomar un ejemplo. La entidad nombrada "Nueva Zelanda" consta de 2 palabras, en un proceso de tokenización clásico, cada palabra estaría presente en el vocabulario. En WordPiece, sin embargo, la palabra "Zelanda" no existe (ya sea porque nunca estuvo presente en el corpus de entrenamiento o porque el algoritmo decidió descartarla porque apareció pocas veces). Luego, para tokenizar el bigrama "Nueva Zelanda", WordPiece devuelve 3 tokens: "Nueva", "Z##" y "##elanda". En este caso, los caracteres "##" indican que el token es en realidad una sub-palabra, y de acuerdo a la posición, indican si la sub-palabra es un pre-fijo (Z##) o un sufijo (##elanda).

[1] `https://huggingface.co/docs/transformers/index`

3.1.2. Replicar los resultados de BETO

En este paso instanciamos la clase *Trainer*[2] de *Hugging Face* y definimos argumentos para el entrenamiento. El objetivo es utilizar esta abstracción y hacer una afinación (fine-tuning) del modelo BETO sobre el corpus de entrenamiento de la tarea **CONLL-2002** buscando replicar los resultados reportados por el grupo que desarrolló BETO (Cañete et al., 2020). La idea es obtener un desempeño igual o superior a lo reportado en la Tabla 2.1 con experimentos locales, para asegurarnos de que el modelo subido a Hugging Face efectivamente funciona y de esa manera poder construir sobre esta base.

3.1.3. Explorar el uso de otros conjuntos de datos para la tarea de reconocimiento de entidades nombradas

Experimentamos con los conjuntos de datos anotados de **WikiNER** (Nothman et al., 2017) y **WikiNEuRal** (Tedeschi et al., 2021) como datos de entrenamiento. La idea es observar el desempeño de estos conjuntos de datos que son más grandes en cantidad de instancias, pero están anotados automáticamente. Utilizamos dichos conjuntos para afinado del modelo de BETO y observamos su desempeño con el conjunto de evaluación de CONLL-2002, medido por el F1-Score. Los resultados de esta etapa sirven para avanzar en la combinación para la creación de un conjunto de entrenamiento combinado entre los datos de CONLL, que tienen menor cantidad pero están anotados manualmente y los datos de WikiNER o WikiNEuRal, de los que hay mayor cantidad pero son más susceptibles al error que surge de la anotación automática.

3.1.4. Combinación de conjuntos de datos de entrenamiento

Buscando poder mejorar el desempeño de nuestros modelos se combinaron los conjuntos de datos de CONLL y WikiNER/WikiNEuRal para hacer afinado del modelo de BETO. Uno de los objetivos es ver si se logra mejorar la métrica de F1-Score sobre el conjunto de evaluación. Sin embargo, no es lo único que nos interesa evaluar.

Parte de utilizar conjuntos de datos de mayor tamaño, aunque estén anotados automáticamente, es la mejora en la cobertura del modelo. La cobertura del modelo se define como la cantidad de pares de Token-Etiqueta, donde la etiqueta es alguna de las ocho clases que representan una entidad nombrada (ya sean B-* o I-*). La idea es que a mayor cobertura, el modelo sabrá reconocer mayor cantidad de ejemplos nuevos a la hora de evaluarse sobre datos no vistos previamente.

Utilizando F1-Score como métrica para el desempeño de los modelos al evaluarlos sobre los datos de CONLL, la combinación de los conjuntos de CONLL y WikiNEuRal no pudo superar el puntaje obtenido al entrenar solo con los datos de CONLL. Pero logró expandir la cobertura del modelo al agregar durante el entrenamiento las nuevas entidades que el conjunto de WikiNEuRal provee.

[2] `https://huggingface.co/docs/transformers/main_classes/trainer`

3.1.5. Estudiar un esquema de entrenamiento semi-supervisado sobre los datos de WikiNEuRal

Finalmente, se explora un método semi-supervisado basado en la idea de "bootstrapping" presentada en (Yarowsky, 1995).

El método consiste en tomar el mejor modelo disponible y utilizarlo para anotar nuevamente el conjunto de datos de WikiNEuRal (que si bien están anotados automáticamente, su desempeño en la evaluación sobre los datos de CONLL no es buena). Con estos datos ya corregidos, se entrena nuevamente el modelo para tener una mejor cobertura y a su vez asegurarnos de no disminuir el desempeño medido sobre la evaluación en la tarea de CONLL.

Empleamos dos formas de realizar el agregado de nuevas instancias. En la primera, se seleccionan los ejemplos cuyas etiquetas coincidan con las predicciones de los nuevos datos; esto quiere decir que se agregan los ejemplos donde coinciden la predicción del modelo de BETO (afinado con los datos anotados manualmente de CONLL), y la anotación automática que se le asignó a la entidad en el conjunto de datos de WikiNEuRal. Y en la segunda iteración de este experimento se agrega la condición de que la etiqueta predicha por el modelo, también tenga un nivel de confianza superior a un umbral dado.

3.2. Conjuntos de datos

En esta sección presentamos los diferentes conjuntos de datos que usamos para el desarrollo de este trabajo, junto con las formas en las cuales trabajamos con cada uno para poder lograr tenerlos de una manera uniforme y de simple acceso.

En el cuadro 3.1 podemos ver en detalle los conjuntos de datos sobre los cuales entrenamos nuestro modelo.

Conjunto de Datos	No. de Oraciones	No. de Palabras
CONLL train	8324	264715
CONLL test	1518	51533
WikiNER	128335	3500013
WikiNeural	76320	1834096

Cuadro 3.1: Conjuntos de datos utilizados

Para hacer uso de estos datos empleamos la funcionalidad de Datasets de Hugging Face[3], la cual nos provee las beneficios descritos al comienzo de la sección.

Afortunadamente, tanto para el caso del conjunto de CONLL[4] y de WikiNEuRal[5], ya tenemos disponibles los conjuntos de datos en el repositorio de Hugging Face y con el formato deseado, por lo cual ya están listos para ser empleados.

Sin embargo, para el caso de WikiNER fue necesario subir el conjunto de datos en Español al repositorio, ya que sólo estaba disponible en el idioma Francés. Tras el proceso descripto en la sección 2.1.2 lo hicimos público en el repositorio y se accede mediante `https://huggingface.co/datasets/NazaGara/wikiner-es`.

[3] `https://huggingface.co/docs/datasets/index`

[4] `https://huggingface.co/datasets/conll2002`

[5] `https://huggingface.co/datasets/Babelscape/wikineural`

Podemos ver en el cuadro 3.2 las diferentes etiquetas representativas que se usan para la tarea de reconocimiento de entidades, junto con un ejemplo de cada una y la cantidad de veces que ocurren en cada conjunto de datos (sumando ambos tipos de etiquetas B-*/I-*).

Tipo Entidad	Ejemplo	Etiquetas	CONLL-2002	WikiNER	WikiNEuRal
LOC	Plaza San Martín	B-LOC, I-LOC, I-LOC	6804	185519	77841
ORG	Universidad de Córdoba	B-ORG, I-ORG, I-ORG	12382	89403	31904
PER	Ricardo Bochini	B-PER, I-PER	8224	143290	73488
MISC	Carta Magna	B-MISC, I-MISC	5385	69509	52969

Cuadro 3.2: Tipos de Entidades

Descartando la clase *'O'*, apreciamos que las distribuciones de las demás clases en los conjuntos generados por CONLL son muy similares, como vemos en la figura 3.1, lo cual es esperable dado que vienen de los mismos datos y fueron generados en simultáneo.

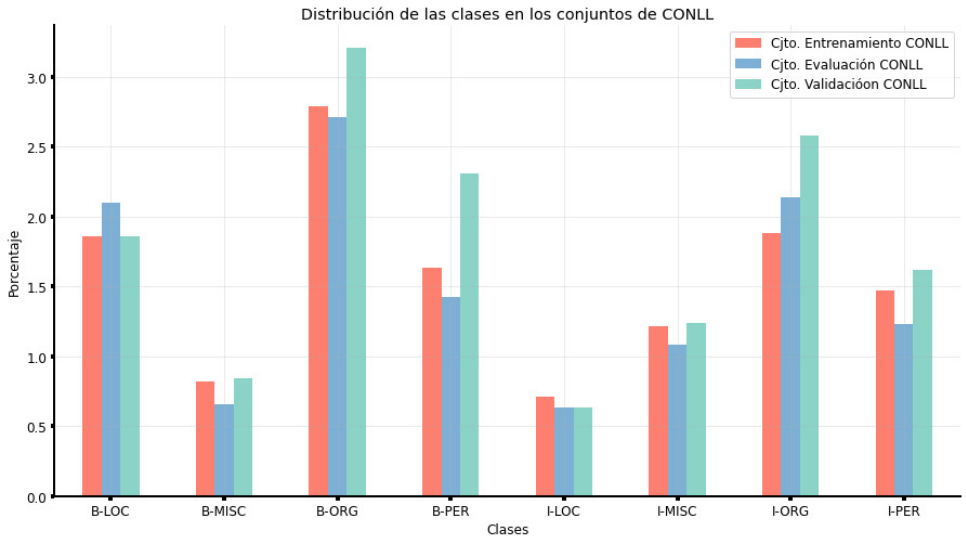


Figura 3.1: Distribución de clases de los conjuntos de CONLL

Mientras que si observamos las distribuciones de las clases de los conjuntos de entrenamiento en el cuadro 3.2, podemos ver que son diferentes en las clases ORG y LOC. Es decir, hay un mayor porcentaje de entidades de tipo Organización en el conjunto de entrenamiento de CONLL que en WikiNER/WikiNEuRal. Mientras que para la clase LOC pasa al revés, hay un mayor porcentaje de nombre de Personas en los conjuntos de datos extraídos de la Wikipedia que en el conjunto de CONLL.

Esto ya nos puede dar una indicación de que estas clases pueden desempeñarse pobremente al momento de evaluar los modelos afinados con los conjuntos de WikiNER y WikiNEuRal, ya que no se asemejan a las distribuciones de los conjuntos de validación ni de evaluación. Mientras que las clases PER y MISC sí lo hacen, con la salvedad de que las entidades Misceláneas dentro del dominio de los textos de la Wikipedia tienen un alcance más grande que en un texto del estilo periodístico.

De la figura 3.2 podemos apreciar la diferencia entre el porcentaje de la clase `'B-ORG'`, lo que nos indica que hay una mayor cantidad de entidades de tipo Organización

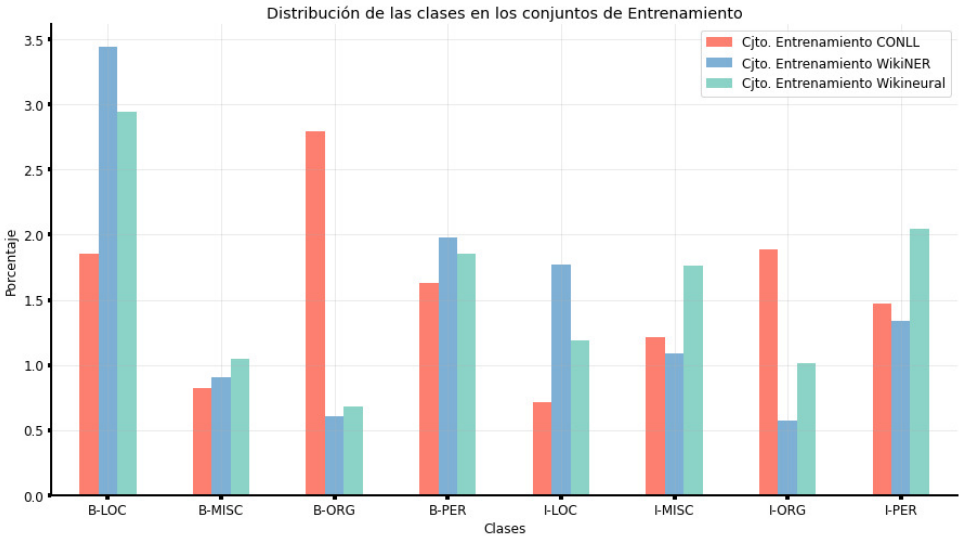


Figura 3.2: Distribución de clases en los conjntos de entrenamiento

en relación a las demás clases en el conjunto de CONLL. Mientras que hay un menor porcentaje de entidades de tipo Lugar que los demás conjuntos de datos basados en la Wikipedia. Vale la pena destacarlo, ya qué esto nos servirá al momento de entrenar el modelo con los distintos datos para poder identificar en que clases el modelo se puede confundir más el momento de evaluar con él conjunto de evaluación de CONLL.

3.3. Entrenamiento

Como describimos previamente, utilizamos la abstracción *Trainer* de Hugging Face para poder entrenar nuestros modelos. Esta es una clase que provee una API de trabajo completa para el entrenamiento de modelos usando *PyTorch*(Paszke et al., 2019). Tiene soporte para entrenamiento en CPU, GPU o TPU, además de la flexibilidad para poder modificar el ciclo de entrenamiento generando subclases para redefinir métodos o mediante los Training Arguments [6] (argumentos de entrenamiento). El modelo que usamos como base para nuestro trabajo es BETO-Cased[7], ya que es importante mantener las letras mayúsculas para reconocer nombres.

Para el entrenamiento y evaluación de los modelos utilizamos tres conjuntos de datos.

- Conjunto de Entrenamiento: se utiliza para entrenar, así el modelo se ajusta para la predicción de nuevos datos.
- Conjunto de Validación: son datos que usan para poder evaluar al modelo durante el entrenamiento y así poder ajustar los hiper-parámetros del modelo.
- Conjunto de Evaluación: solo se utiliza para evaluar el modelo y deben ser datos que el modelo no haya usado para entrenar. Sobre este conjuntos se reportan los datos en el capítulo 4.

[6] `https://huggingface.co/docs/transformers/main_classes/trainer#transformers.TrainingArguments`

[7] `https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased`

3.4. Decisiones de Diseño

3.4.1. Padding

Para poder entrenar el modelo, a través de la tokenización de sub-palabras traducimos cada palabra a múltiples tokens, de los cuales el modelo posee una representación vectorial (embedding) de cada uno. Estos tokens se procesan en simultáneo y cada uno influye a los adyacentes, lo que permite al modelo aprender del contexto de cada palabra. Los tokens se organizan dentro de una secuencia, resultando cada oración en una lista de vectores. Pero necesitamos que cada lista tenga la misma cantidad de elementos para usar de manera eficiente el poder de cómputo de la GPU, y dado que las oraciones no tienen necesariamente la misma longitud, se realiza un relleno con un token especial que le indica al modelo que ese token que se añade no debe considerarse para el entrenamiento del modelo. A este proceso se lo denomina *padding* o relleno.

La decisión llegó al momento de adoptar qué estrategia utilizar para el padding. Tras algunas pruebas tomamos la estrategia `longest on batch`, es decir, que se tomaba la longitud de la oración más grande de cada subconjunto de entrenamiento (o "batch" en inglés) y de forma dinámica se realizaba el relleno de los demás ejemplos con el token especial de padding. Esta decisión repercute al momento de tomar métricas, ya que debemos eliminar de cada ejemplo las predicciones que se correspondan a los lugares donde se agrego el token usado para rellenar.

Por ejemplo, si aplicamos la estrategia `longest on batch` y lo usamos para las frases: Ignacio hizo un viaje por la ciudad de Buenos Aires. Vicente del Bosque es un ex jugador de fútbol nacido en España. El relleno deja las frases como se muestra en la tabla 3.3.

numero Frase	item 0	item 1	...	item 11	item 12	item 13	item 14
1	[CLS]	Ignacio	...	.	[SEP]	[PAD]	[PAD]
2	[CLS]	Vicente	...	en	España	.	[SEP]

Cuadro 3.3: Ejemplo de relleno de frases

- `[CLS]` que identifica el comienzo del ejemplo y el tipo de tarea a realizar, en este caso clasificación.
- `[SEP]` que identifica el final del ejemplo.
- `[PAD]` representa el relleno que se añade.

3.4.2. Ajuste de Etiquetas

Como mencionamos en la sección 3.1, utilizamos Tokenización por sub-palabras, para poder entrenar en base a los tokens que el modelo reconoce. Decidimos tomar la opción de replicar la etiqueta de una palabra para cada una de las sub-palabras que conoce el modelo.

En esto nos diferenciamos del trabajo original de BETO (Cañete et al., 2020), ya que ellos decidieron asignar un valor especial para la etiqueta de una sub-palabra que no sea la primera. Nuestra decisión de replicar la etiqueta en lugar de utilizar una etiqueta nueva, fue para facilitar el proceso de evaluación y evitar tener que lidiar con nuevas clases.

En la figura 3.3 continuamos el ejemplo del proceso de Tokenización por subpalabras, mostrando el ajuste de etiquetas que realizamos.

Figura 3.3: Ejemplo de tokenización por subpalabras y réplica de etiquetas

3.4.3. Depuración de Oraciones

Para poder evitar el problema de overfitting del modelo fue necesario remover de los conjuntos de datos aquellos ejemplos que no contengan entidades nombradas, ya que no son útiles para el entrenamiento y de no hacerlo los modelos convergían a una solución incorrecta.

Esta decisión se dio luego de que en las primeras pruebas que realizamos, dado que el conjunto de datos original de CONLL tiene más de 2100 oraciones en las cuales no hay ninguna entidad nombrada (las cuales representan un 25 % del conjunto) y el modelo convergía rápidamente a la solución de que ningún ejemplo tenia entidades.

Capítulo 4

Experimentos y Análisis de Resultados

En este capítulo presentamos los experimentos que realizamos, incluyendo que los conjuntos de datos que se utilizaron en cada uno y un análisis de los resultados obtenidos. Todas las métricas fueron resultado de evaluar los modelos con el conjunto de datos de evaluación de CONLL-2002.

4.1. Experimentos con diferentes conjuntos de entrenamiento

En esta sección se busca comparar los distintos conjuntos de entrenamiento que se presentan en la Sección 3.2. Cada conjunto se utiliza por separado como base para afinar el modelo de BETO y se evalúan sobre la tarea de CONLL-2002. Además de utilizar cada conjunto de datos, también se utilizó el modelo de BETO sin ningún tipo de afinación para tener una referencia de cuánto mejora cada entrenamiento en comparación.

Para cada conjunto de datos probamos diferentes hiper-parámetros de tasa de aprendizaje (learning rate) y cantidad de épocas de entrenamiento sobre el *Trainer*, buscando encontrar aquellos con el mejor desempeño sobre el conjunto de datos de validación. Los hiper-parámetros finales de cada modelo están dados en la Tabla 4.1. Para el resto de los hiper-parámetros del modelo se dejaron los valores fijos que vienen por defecto, ya que suelen ser los mejores.

Hiper-Parámetro	Sin afinación	CONLL-2002	WikiNER	WikiNEuRal
Tasa de aprendizaje	0	3e−5	1e−5	1e−5
Número épocas	0	3	2	3

Cuadro 4.1: Hiper-parámetros de los primeros experimentos

Una vez determinados los mejores hiper-parámetros se evaluaron los modelos sobre el conjunto de datos de evaluación de CONLL, y se reportan los resultados de la métrica F1-Score para cada clase en la Tabla 4.2. Como se puede observar en el cuadro, en primer lugar, cualquier conjunto de datos para afinación es mejor que no hacer ningún tipo de

afinación, esto es esperable, pero es bueno remarcarlo para entender que BETO por si mismo no es útil sin algún tipo de proceso de afinado.

Clase	Sin afinación	CONLL-2002	WikiNER	WikiNEuRal
O	0,169	0,997	0,992	0,989
B-PER	0,088	0,974	0,925	0,914
I-PER	0,027	0,981	0,947	0,958
B-ORG	0,020	0,906	0,749	0,721
I-ORG	0,038	0,911	0,638	0,827
B-LOC	0,085	0,887	0,725	0,803
I-LOC	0,002	0,854	0,448	0,710
B-MISC	0,033	0,788	0,412	0,433
I-MISC	0,026	0,857	0,511	0,602

Cuadro 4.2: Métricas para cada clase afinando BETO con distintos corpus de entrenamiento (además del resultado sin afinar BETO).

En segundo lugar, es claro que la anotación manual que posee el conjunto de datos de CONLL, además de la similitud en la distribución entre los dominios de entrenamiento y evaluación, hace que el modelo entrenado con dichos datos sea mejor que los demás, independientemente de la cantidad de ejemplos que cada conjunto tenga. Para poder visualizar una comparación más cercana a los resultados obtenidos en el trabajo original de BETO (Cañete et al., 2020), podemos observar un promedio no pesado en la Tabla 4.3.

Conjunto entrenamiento	Precision	Recall	F1-Score
Sin afinación	0,101	0,136	0,054
CONLL-2002	**0,986**	**0,900**	**0,906**
WikiNER	0,958	0,726	0,705
WikiNEuRal	0,965	0,757	0,773

Cuadro 4.3: Métricas de las pruebas.

No obstante, como se explicó en el Capítulo 3, la ventaja de utilizar modelos anotados automáticamente reside en la cobertura de dichos modelos. Esto es la cantidad de ejemplos que los corpus poseen por clases. Esto se reporta en la Tabla 4.4, donde se observa la gran cantidad de entidades con las que cuentan los conjuntos de datos basados en la Wikipedia.

Conjunto entrenamiento	Cobertura
CONLL-2002	9640
WikiNER	54922
WikiNEuRal	42300

Cuadro 4.4: Cobertura de cada conjunto de datos.

4.1.1. Discusión sobre los primeros resultados

Como recordamos en la sección 3.2, la figura 3.2 nos muestra las diferencias entre las distribuciones de entidades de los distintos conjuntos de entrenamiento. Recordemos que había dos clases que tenían una distribución similar en los 3 conjuntos de entrenamiento, PER y MISC, mientras que las otras dos, ORG y LOC, tenían una diferencia distribucional bastante marcada. De allí podíamos hipotetizar que utilizar los conjuntos WikiNER y WikiNEuRal podría ser beneficioso sólo sobre aquellas clases que tuviesen una distribución similar. Vemos, sin embargo, que este no es el caso, y la única clase cuyo desempeño se mantiene más o menos similar (o al menos no cae tanto en comparación) a lo obtenido con CONLL es el caso de la clase para personas. Esto dio pie a hacer un análisis más fino de lo que está pasando, particularmente con la clase MISC, que tiene una distribución similar en los conjuntos de entrenamiento pero cuyo desempeño es mucho peor utilizando otros conjuntos que no sean CONLL.

Las figuras 4.1, 4.2 y 4.3 nos muestran las matrices de confusión sobre los datos de evaluación para los modelos afinados con CONLL, WikiNER y WikiNEuRal respectivamente. Por un lado, como originalmente pensábamos que iba a suceder, hay cierta confusión, en particular del modelo entrenado con WikiNEuRal, con las etiquetas de ORG y LOC. Nuevamente, esto debido a la gran diferencia entre las distribuciones de dichas etiquetas entre corpus. Por otro lado, hay una gran pérdida de desempeño para MISC para los modelos que no son CONLL, la cuál es clasificada en muchos casos como LOC y en otros tantos como O. Esto puede atribuirse a la diferencia de dominios de los datos de entrenamiento. Mientras que el CONLL es principalmente de un dominio periodístico, Wikipedia tiene un dominio de datos mucho mayor, luego la clase MISC, que es la más ambigua, puede tener ejemplos mucho más variados, lo que hace difícil su evaluación sobre el corpus de CONLL.

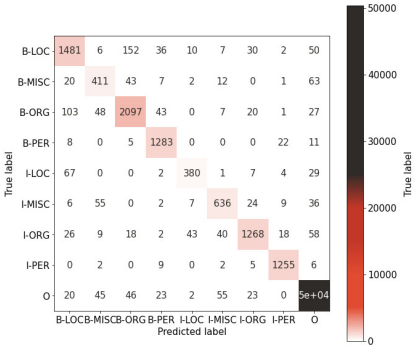


Figura 4.1: CONLL
Matriz de confusión

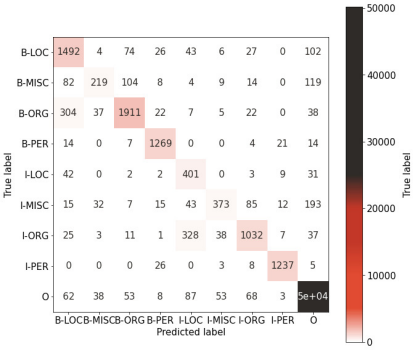


Figura 4.2: WikiNER
Matriz de confusión

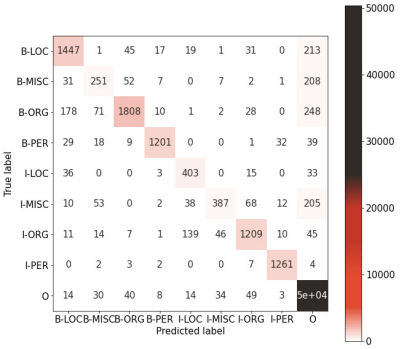


Figura 4.3: WikiNEuRal
Matriz de confusión

Tomamos como ejemplo la frase del conjunto de validación de CONLL-2002: `Los contratos futuros del crudo Brent bajaron hoy, martes, en el mercado de Londres`. Podemos ver en el cuadro 4.5 las diferentes predicciones para las entidades que están en la frase para cada uno de los modelos entrenados.

Notar que al predecir con el modelo entrenado con WikiNER/WikiNEuRal obtenemos la etiqueta *B-ORG* para la palabra 'Brent', cuando la etiqueta verdadera es *B-MISC* en el conjunto de datos de CONLL, y es correctamente predicha por el modelo entrenado con

Palabra	Predicción CONLL-2002	Predicción WikiNER	Predicción WikiNEuRal
Brent	'B-MISC'	'B-ORG'	'B-ORG'
Londres	'B-LOC'	'B-LOC'	'B-LOC'

Cuadro 4.5: Confusión en la predicción de las entidades

CONLL, esto ya que en la Wikipedia no existe el artículo 'Brent' por si solo, como si existe el artículo *Petróleo Brent*, además de que hay ejemplos en el conjunto de entrenamiento de CONLL que lo nombran de esa manera.

Otro caso similar de la ambigüedad de MISC, también obtenido a partir de los datos de validación: `...como ya había evidenciado en oportunidades tan claras que tuvo igualmente en Las Fallas de Valencia y La Magdalena de Castellón`. Para las frases: "Las Fallas de Valencia" y "La Magdalena de Castellón" podemos ver el resultado de las predicciones en el cuadro 4.6. Los tres modelos predicen que son ubicaciones de las ciudades españolas, pero en realidad las frases hablan de dos fiestas, una en cada ciudad. En este caso ningún modelo logra capturar la diferencia entre las entidades, lo cual nos indica que hay ciertos ejemplos que son difíciles de clasificar independientemente del conjunto de entrenamiento utilizado.

Palabra	Etiqueta verdadera	Predicción CONLL-2002	Predicción WikiNER	Predicción WikiNEuRal
Las	'B-MISC'	'B-LOC'	'B-LOC'	'B-LOC'
Fallas	'I-MISC'	'I-LOC'	'I-LOC'	'I-LOC'
de	'O'	'I-LOC'	'I-LOC'	'I-LOC'
Valencia	'B-LOC'	'I-LOC'	'I-LOC'	'I-LOC'
y	'O'	'O'	'O'	'O'
La	'B-MISC'	'B-LOC'	'B-LOC'	'B-LOC'
Magdalena	'I-MISC'	'I-LOC'	'I-LOC'	'I-LOC'
de	'O'	'O'	'I-LOC'	'I-LOC'
Castellón	'B-LOC'	'B-LOC'	'I-LOC'	'I-LOC'

Cuadro 4.6: Errores en predicciones para los modelos

4.2. Optimizando Conjuntos de datos

Dado que los resultados obtenidos en los modelos entrenados previamente no se lograron desempeñar de la mejor forma, decidimos combinar los conjuntos de datos para poder aumentar la métrica para las clases de peor desempeño. Decidimos experimentar en estas pruebas con el conjunto de WikiNEuRal ya que este obtuvo mejores resultados en comparación con el modelo entrenado con WikiNER. Tomamos dos enfoques en esta etapa.

4.2.1. Manteniendo distribuciones originales de CONLL

Consiste en extender el conjunto de entrenamiento de CONLL con ejemplos de WikiNEuRal, intentando mantener las distribuciones originales de CONLL que vemos en

la figura 3.1, ya que con estas distribuciones el modelo resultante obtuvo los mejores resultados.

Para combinar los conjuntos, probamos con quedarnos únicamente con los ejemplos que contengan alguna de las etiquetas *'B-ORG'* (y por consecuente podían tener la etiqueta *'I-ORG'*) o quedarnos con los ejemplos que también contengan *'B-MISC'*.

En caso de que una oración no cumpla el requisito, tiene una probabilidad del 5 %[1] de ser agregada. El valor fijado nos permite seguir aumentando la cantidad total de ejemplos de entrenamiento, mientras nos mantenemos cercanos a las distribuciones de CONLL. También probamos hacer un conjunto de validación híbrido usando los conjuntos de validación de WikiNEuRal y de CONLL para intentar obtener mejores parámetros de entrenamiento que se vieran reflejados en los resultados de evaluación.

En las tablas 4.7 y 4.8 podemos ver los resultados obtenidos para estos experimentos, donde en la primera se detalla el procesamiento realizado y en la segunda tabla se muestran los valores de la métrica F1-Score para cada clase.

n° experimento	Nombre	Validación	F1-Score
Experimento 1	100 % CONLL+50 % WikiNEuRal ORG y MISC	CONLL	90,76
Experimento 2	100 % CONLL+50 % WikiNEuRal ORG	CONLL	89,99
Experimento 3	100 % CONLL+50 % WikiNEuRal ORG	Ambos	89,71

Cuadro 4.7: Pruebas manteniendo distribuciones.

Podemos ver en la figura 4.4 como quedaron las distribuciones para estos experimentos intentando mantener la distribución de clases originales que tenia CONLL. Notar que tanto para los Experimentos 2 y 3 se utilizo el mismo conjunto de entrenamiento y solo vario el conjunto de validación.

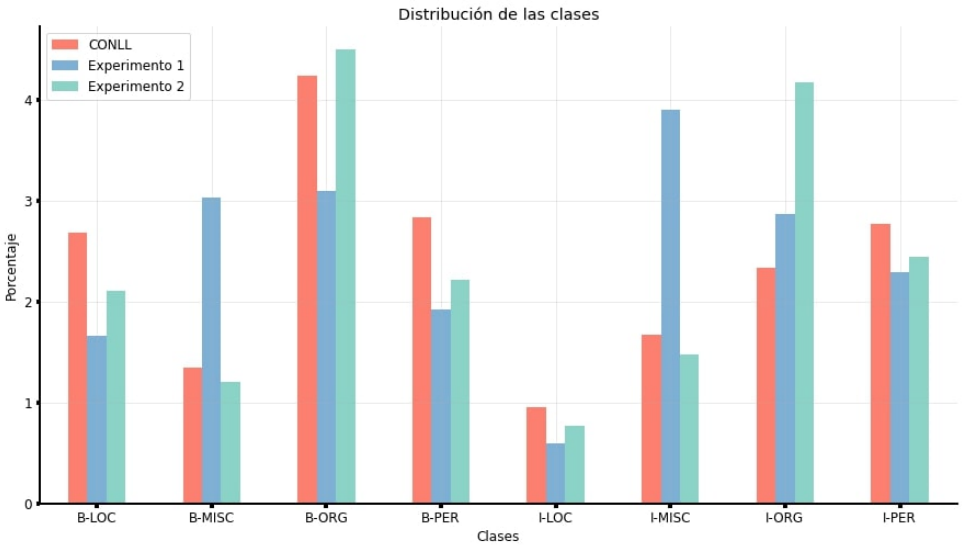


Figura 4.4: Distribución de clases

[1]Si bien no hay ninguna justificación teórica para el valor, comprobamos en diferentes pruebas que es un valor que funcionaba bien

Clase	Experimento 1	Experimento 2	Experimento 3
O	0,997	0,997	0,996
B-PER	0,975	0,969	0,973
I-PER	0,981	0,979	0,976
B-ORG	0,911	0,908	0,899
I-ORG	0,915	0,911	0,910
B-LOC	0,897	0,894	0,887
I-LOC	0,869	0,868	0,856
B-MISC	0,772	0,741	0,740
I-MISC	0,852	0,834	0,837

Cuadro 4.8: Métricas para cada clase.

4.2.2. Extendiendo las clases que funcionaron bien

Cómo podemos ver en la tabla 4.2, el modelo que fue refinado con los datos de WikiNEuRal obtuvo buenos resultados para las clases *'PER'* y *'LOC'* por lo cual probamos con extender el conjunto de entrenamiento de CONLL con los ejemplos que contengan alguna de estas clases, seleccionando oraciones de la forma descrita en la sub-sección anterior pero con estas nuevas clases. También, cómo los experimentos nombrados anteriormente, probamos con combinar los conjuntos de validación de ambas fuentes.

En la tablas 4.9 y 4.10 podemos ver los resultados obtenidos para estos experimentos, en la primera identificando a cada experimento junto con el valor macro del resultado, y en la segunda viendo una por una el desempeño de cada clase.

n° experimento	Nombre	Validación	F1-Score
Experimento 4	100 % CONLL+50 % WikiNEuRal PER	CONLL	90,89
Experimento 5	100 % CONLL+50 % WikiNEuRal PER y LOC	CONLL	89,89

Cuadro 4.9: Resultados para de pruebas extendiendo clases Persona y Lugar.

Clase	Experimento 4	Experimento 5
O	0,997	0,997
B-PER	0,970	0,969
I-PER	0,979	0,977
B-ORG	0,918	0,901
I-ORG	0,921	0,914
B-LOC	0,891	0,884
I-LOC	0,859	0,851
B-MISC	0,783	0,753
I-MISC	0,863	0,846

Cuadro 4.10: Métricas para cada clase.

4.2.3. Discusión de experimentos combinando datos

Lo primero que se puede observar es que en los experimentos realizados los resultados no varían significativamente del obtenido al entrenar puramente con CONLL. Esto reafirma la hipótesis de que incrementar la cantidad de oraciones no aumenta necesariamente el desempeño. Además de que en ninguno de los experimentos se logra mejorar la métrica para la clase a la cual se apuntó, aunque sí hay algunos resultados interesantes medidos desde el punto de desempeño.

Por ejemplo, para el Experimento 4 el objetivo fue el de aumentar las ejemplos de entrenamiento con aquellas que contengan al menos una entidad de tipo *'PER'*, sin embargo esto no se refleja en una mejora en el valor de la métrica para dicha clase. No obstante, sí hay una mejora para las clases ORG y LOC, que eran unas de las clases donde el modelo entrenado con WikiNEuRal tiene algunas confusiones. Más aún, el hecho de agregar más datos de LOC claramente tiene un resultado positivo porque hay una mejora del desempeño con respecto a lo que se logra entrenando el modelo únicamente con CONLL, y esto puede deberse en parte a la mayor cobertura que agregan los ejemplos nuevos de WikiNEuRal, que posee mucha mayor cantidad de ejemplos del tipo LOC (tanto absolutos como relativos) respecto a la distribución de dichas entidades en CONLL.

Pero también en estos experimentos la clase que impide que el F1-Score aumente sigue siendo la Miscelánea, la cual al intentar aumentar la cantidad de sentencias que contengan esta clase solo hace que el modelo se confunda más para predecir dicha clase, nuevamente se estima que esto es resultado de la diferencia de dominios entre CONLL y Wikipedia.

Para poder tomar una medida cualitativa del desempeño de los nuevos modelos, volvemos al ejemplo presentado en la sección anterior. En el cuadro 4.11 podemos ver las predicciones de los modelos de los experimentos 1 y 4 para dicho ejemplo. Seleccionamos estos experimentos ya que en el Experimento 1 hacemos hincapié en agregar entidades de tipo Miscelánea y en el Experimento 4 ya que fue el que mejor desempeño tuvo.

Palabra	Etiqueta verdadera	Predicción Experimento 1	Predicción Experimento 4
Las	'B-MISC'	'B-LOC'	'B-LOC'
Fallas	'I-MISC'	'I-LOC'	'I-LOC'
de	'O'	'O'	'I-LOC'
Valencia	'B-LOC'	'I-LOC'	'I-LOC'
y	'O'	'O'	'O'
La	'B-MISC'	'B-LOC'	'B-LOC'
Magdalena	'I-MISC'	'I-LOC'	'I-LOC'
de	'O'	'O'	'I-LOC'
Castellón	'B-LOC'	'B-LOC'	'I-LOC'

Cuadro 4.11: Predicciones de los modelos para el ejemplo.

Podemos ver que el Experimento 1 aprendió a diferenciar las ciudades "Valencia" y "Castellón" pero aún no llega a determinar las entidades como fiestas y consecuentemente, como entidades Misceláneas. Mientras que en el Experimento 4, y hasta ahora el de mayor desempeño en general, no logra distinguir las entidades.

4.3. Entrenamiento semi-supervisado utilizando bootstrap

Como se explicó en la sub-sección 3.1.5, en esta instancia de trabajo se buscó entrenar un modelo semi-supervisado basado en el algoritmo de bootstrapping (Yarowsky, 1995). Para dicho modelo, es necesario un conjunto de entrenamiento inicial supervisado y un conjunto de predicción no supervisado de donde se obtendrán los ejemplos que se irán agregando al modelo en cada iteración de bootstrap. Por la distribución de los datos de evaluación, se decidió utilizar WikiNEuRal como el conjunto de datos no supervisado. A la hora de elegir el modelo inicial sobre el que bootstrap será construido, se decidió utilizar exclusivamente los datos de CONLL. Si bien los experimentos hechos hasta el momento indican una leve mejora en el desempeño con los datos como se establecen en el experimento 4 del cuadro 4.9, el hecho de que la clase MISC pierda desempeño como indica el cuadro 4.10 es indicativo de que eso puede arrastrarse y terminar afectando gravemente en el proceso de bootstrapping. Además, de esta forma se evita que durante el proceso de predicción, que es sobre WikiNEuRal, haya un solapamiento entre los datos de entrenamiento y los nuevos datos que son agregados luego de cada iteración de bootstrap.

Una vez obtenidas las predicciones, planteamos dos formas diferentes para seleccionar qué nuevos ejemplos usar para entrenar.

4.3.1. Bootstrapping viendo predicciones

La primer forma de Bootstrapping que desarrollamos consiste en predecir las etiquetas y compararlas con las etiquetas que tenían originalmente asignadas en el corpus. De esta forma, evaluamos cada ejemplo en el conjunto de WikiNEuRal y comparamos palabra por palabra las predicciones obtenidas con la etiqueta original. En caso de que no coincida la predicción con la etiqueta, se descartaba toda la oración.

En las tablas 4.12 y 4.13 podemos ver los resultados de los experimentos bajo el primer esquema de Bootstrapping.

No. experimento	Nombre	F1-Score
Experimento 6	Bootstrap 50 % WikiNEuRal	88,45
Experimento 7	Bootstrap 100 % WikiNEuRal	88,81

Cuadro 4.12: Resultados del primer método de Bootstrap

Podemos apreciar que esta forma de aplicar el Bootstrapping empeora los resultados obtenidos ya que decae el rendimiento de las clases Misceláneas.

4.3.2. Bootstrapping con un criterio de confianza

El segundo método consiste en utilizar el primer método, pero observando con mayor detalle las predicciones obtenidas. Ayudándonos con una Función Softmax [2] vemos con qué porcentaje se corresponde cada clase en cada palabra.

[2] `https://en.wikipedia.org/wiki/Softmax_function`

Clase	Experimento 6	Experimento 7
O	0,996	0,996
B-PER	0,966	0,965
I-PER	0,979	0,979
B-ORG	0,898	0,901
I-ORG	0,901	0,910
B-LOC	0,885	0,882
I-LOC	0,843	0,847
B-MISC	0,715	0,718
I-MISC	0,783	0,796

Cuadro 4.13: Métricas para cada clase en los experimentos de bootstrap por predicciones.

Así podemos quedarnos sólo con aquellas predicciones que sean mayores a un cierto valor, ganando una noción de criterio de confianza en las predicciones. Seguimos este procedimiento para cada palabra en una oración, y si en alguna predicción no tenemos un porcentaje mayor al fijado descartábamos toda la oración.

En las tablas 4.14 y 4.15 mostramos el rendimiento de los experimentos de este método de hacer el Bootstrap. En comparación con la forma descrita en la sección 4.3.1, en esta obtenemos un resultado más similar al obtenido cuando entrenamos con el conjunto de datos de CONLL.

No. experimento	Nombre	F1-Score
Experimento 8	Bootstrap 95 % a 50 % WikiNEuRal	89,41
Experimento 9	Bootstrap 95 % a 100 % WikiNEuRal	90,00
Experimento 10	Bootstrap 99 % a 50 % WikiNEuRal	89,40
Experimento 11	Bootstrap 95 % a WikiNER+WikiNEuRal	89,60

Cuadro 4.14: Resultados del segundo método de Bootstrap

Clase	Experimento 8	Experimento 9	Experimento 10	Experimento 11
O	0,997	0,997	0,996	0,996
B-PER	0,962	0,964	0,966	0,968
I-PER	0,975	0,977	0,976	0,978
B-ORG	0,902	0,906	0,900	0,904
I-ORG	0,909	0,905	0,909	0,910
B-LOC	0,886	0,887	0,888	0,886
I-LOC	0,847	0,859	0,836	0,841
B-MISC	0,741	0,783	0,729	0,757
I-MISC	0,828	0,822	0,846	0,825

Cuadro 4.15: Métricas para cada clase en los experimentos de Bootstrap utilizando un nivel de confianza.

4.3.3. Discusión del método de Bootstrapping

A simple vista, ninguno de los métodos de Bootstrapping resultó en una mejora significativa del desempeño del modelo, lo cual es algo que se viene repitiendo en los experimentos. Pero logramos mantener el desempeño de la métrica, mientras aumentamos significativamente la cobertura del modelo como vemos en el cuadro 4.16.

Experimento	Cobertura
CONLL Sólo	9640
Experimento 6	27597
Experimento 7	37766
Experimento 8	14935
Experimento 9	18599
Experimento 10	13913
Experimento 11	23237

Cuadro 4.16: Cobertura en los experimentos.

Gracias al proceso de Bootstrapping el modelo ahora aprendió sobre nuevas etiquetas para palabras que ya conocía, por ejemplo, dentro del conjunto de datos de entrenamiento de CONLL se habla de 'Sydney' como la ciudad de Australia, por lo que está etiquetada como *'B-LOC'*. Pero luego del procedimiento de bootstrapping utilizando la totalidad de los datos de WikiNEuRal, el modelo también se entreno con el par 'Sydney'-*'I-MISC'* ya que se hace referencia a los Juegos Olímpicos de Sydney del año 2000.

Con esto logramos aumentar los contextos en los cuales aparecen diferentes palabras, aumentando los casos de uso del modelo. Si tomamos la frase: `La selección Estadounidense de baloncesto se colgó la medalla de oro en los juegos de Sydney 2000, con un plantel de jugadores NBA`. En el cuadro 4.17 podemos ver las etiquetas con la que los modelos predicen parte de la frase.

Palabra	Predicción CONLL	Predicción Experimento 11
los	'O'	'O'
juegos	'O'	'O'
de	'O'	'O'
Sydney	'B-LOC'	'B-MISC'
2000	'I-MISC'	'I-MISC'

Cuadro 4.17: Predicciones de los modelos para el ejemplo.

Ya que ahora el modelo conoce nuevos contextos en los cuales aparece la entidad 'Sydney', es capaz de predecir correctamente del resto de la frase que se habla de los Juegos Olímpicos y no de la ciudad australiana.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

En este trabajo presentamos diferentes enfoques para intentar mejorar el desempeño de BETO, un modelo de tipo BERT, para la tarea del reconocimiento de entidades nombradas en el idioma Español. De la investigación realizada podemos determinar que el mejor modelo se obtuvo cuando se entreno al modelo con los datos creados para la competencia de CONLL en 2002 (Tjong Kim Sang, 2002), donde obtuvimos un puntaje competitivo con respecto a otros trabajos dentro del idioma.

La problemática más importante para la tarea de reconocimiento de entidades es la poca cantidad de datos correctamente etiquetados, y hasta que no estén disponibles más conjuntos de datos en el idioma Español será muy difícil poder mejorar el desempeño de los modelos. Mientras tanto, podemos utilizar conjuntos de datos anotados de forma semi-automática para expandir dichos modelos. Estos datos, si bien no garantizan una mejora del desempeño, sí brindan un desempeño comparable, y permiten incrementar considerablemente la cantidad de entidades que el modelo reconoce, dejando de lado las barreras entre los dominios del texto.

Destacamos también que el uso de los corpus extraídos de la Wikipedia no se desempeñan bien para predecir textos menos formales como lo son los artículos periodísticos o conversaciones casuales. Además de que el tipo de anotación realizado en los trabajos de WikiNER (Nothman et al., 2017) y WikiNEuRal(Tedeschi et al., 2021) se aprovecha de la clasificación en entidades de los artículos, lo que arrastra un grado de error ya que sólo conoce las entidades que tengan un hipervínculo válido dentro de Wikipedia.

También pudimos ver como las abstracciones disponibles en Hugging Face hacen que sea relativamente sencillo poder introducirse al desarrollo de modelos del lenguaje y que el modelo existente de BETO es un modelo prometedor con respecto a las diversas tareas del Procesamiento del Lenguaje Natural actuales, y que es una arquitectura que tiene mucho para ofrecer en el desarrollo de herramientas para el lenguaje.

5.2. Trabajo Futuro

Como primer instancia de trabajo futuro, recalcamos el uso los datos que se generaron para la competencia de CAPITEL en 2020(Porta and Espinosa-Anke, 2020) haciendo el

mismo procedimiento que realizamos, en el momento en que estos nuevos datos estén públicos para su uso. Dado que estos son datos que fueron anotados a mano y revisados, consideramos que la calidad de estos datos, al combinarse con los demás datos de CONLL-2002 pueden aumentar el desempeño del modelo.

Dentro de los formatos desarrollados para la anotación de tokens, además del formato *IOB* ya mencionado, existen otros esquemas de anotación, los cuales pueden ayudar a un modelo a mejorar su desempeño (Alshammari and Alanazi, 2021) y pueden ser evaluados dentro del idioma Español.

Otra reciente investigación se propuso evaluar los modelos de reconocimiento de entidades considerados 'state-of-the-art', estos son aquellos que poseen actualmente el mejor desempeño al momento de hacerse público. En dicho trabajo (Vajjala and Balasubramaniam, 2022) se plantean nuevas formas de evaluación para estos modelos, que podríamos usar para determinar de una manera más objetiva el desempeño de los modelos y evaluar también los conjuntos de datos que se utilizan.

Bibliografía

Alshammari, N. and Alanazi, S. (2021). The impact of using different annotation schemes on named entity recognition. *Egyptian Informatics Journal*, 22(3):295–302.

Cañete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H., and Pérez, J. (2020). Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan. Association for Computational Linguistics.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition.

Li, Y., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., and Jagadish, H. (2008). Regular expression learning for information extraction. pages 21–30.

Magnolini, S., Piccioni, V., Balaraman, V., Guerini, M., and Magnini, B. (2019). How to use gazetteers for entity recognition with neural models. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 40–49, Macau, China. Association for Computational Linguistics.

Morwal, S., Jahan, N., and Chopra, D. (2012). Named entity recognition using hidden markov model (hmm). *International Journal on Natural Language Computing*, 1:15–23.

Nothman, J., Ringland, N., Radford, W., Murphy, T., and Curran, J. R. (2017). Learning multilingual named entity recognition from Wikipedia.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pires, T., Schlinger, E., and Garrette, D. (2019). How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Porta, J. and Espinosa-Anke, L. (2020). Overview of capitel shared tasks at iberlef 2020: Named entity recognition and universal dependencies parsing.

Ramshaw, L. A. and Marcus, M. P. (1995). Text chunking using transformation-based learning.

Schuster, M. and Nakajima, K. (2012). Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Sundheim, B. M. (1995). Overview of results of the muc-6 evaluation. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, page 13–31, USA. Association for Computational Linguistics.

Tedeschi, S., Maiorca, V., Campolungo, N., Cecconi, F., and Navigli, R. (2021). WikiNEuRal: Combined neural and knowledge-based silver data creation for multilingual NER. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tjong Kim Sang, E. F. (2002). Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Vajjala, S. and Balasubramaniam, R. (2022). What do we really know about state of the art ner?

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.