Creación eficiente de modelos estadísticos para detección automática y precisa de entidades nombradas

Horacio Miguel Gómez $(L : 50825)$
Juan Pablo Orsay $(L : 49373)$

Proyecto final de carrera

2019-11-20

Índice

1. Introducción

El **Reconocimiento de Entidades Nombradas** (*NER*) es una subtarea de extracción de información que busca ubicar las menciones de **entidades nombradas** en textos no estructurados. Estas entidades son luego clasificadas en categorías predefinidas como los nombres de personas, organizaciones, ubicaciones, expresiones de tiempo, cantidades, valores monetarios, porcentajes, entre otras.

Por ejemplo en el siguiente texto:

> Este es el proyecto final de carrera de los alumnos Gómez y Orsay para el Instituto Tecnológico de Buenos Aires.

Se pueden detectar 3 entidades:

- **Gómez**: Persona
- **Orsay**: Persona
- **Instituto Tecnológico de Buenos Aires**: Organización

El estado-del-arte de los sistemas *NER* producen un rendimiento casi humano(Marsh & Perzanowski, 1998) (cercanos al 95 % de *valor-F*).

A pesar de estos altos valores de rendimiento, la industria tiene dificultades para poder capitalizar la efectividad de dichas sistemas y algoritmos. Es por ello que en este trabajo final hemos tomado la decisión de implementar una plataforma Open Source para revertir esta situación.

2. Definición del problema

2.1. Conceptos básicos

2.1.1. Entidad nombrada

En *extracción de información*, una *entidad nombrada* es un objeto del mundo real como lo son personas, ubicaciones, organizaciones, productos, etc; que pueden denotarse con un nombre propio. La entidad puede ser abstracto o tener existencia física. Ejemplos de entidades nombradas son «Mauricio Macri», «Ciudad Autónoma de Buenos Aires», «Apple Macbook». También se suele definir sencillamente como aquellas entidades que se pueden ver como instancias de entidad (por ejemplo, la Ciudad Autónoma de Buenos Aires es una instancia de una ciudad).

2.1.1.1. Definición formal

Formalmente el concepto de «entidad nombrada» se deriva de la definición del filósofo estadounidense **Saul Kripke** de **designador rígido** (Kripke, 1980) que forma parte de la lógica modal y filosofía del lenguaje.

> Un designador rígido designa a una misma entidad en todos los mundos posibles en los que esa entidad existe, y no designa nada en aquellos mundos en los que no existe.

Algunos ejemplos de designadores rígidos son:

- Nombres propios como «Saul Kripke», «Júpiter», «Londres», «4» y «Hércules».
- Descripciones definidas matemáticas como «la raíz cuadrada de 4» y «8 - 2».
- Nombres de clases naturales como «agua» y «bronce».
- Nombres de sensaciones como «dolor» y «alegría».

Por el contrario, los **designadores flácidos** pueden designar diferentes cosas en diferentes mundos posibles y **no** son *entidades nombradas*.

Por ejemplo en la oración: «Mauricio Macri es el presidente de Argentina»:

- «Mauricio Macri» y «Argentina» son *entidades nombradas*, ya que se refieren a objetos específicos.
- «presidente» y «presidente de Argentina» no son *entidades nombradas*, ya que pueden usarse para referirse a muchos objetos diferentes en mundos diferentes:
  - «presidente» puede ser de diferentes países u organizaciones que se refieren a diferentes personas.
  - «presidente de Argentina», si bien hace referencia a un mismo pais, puede ser de diferentes períodos presidenciales que se refieren a diferentes personas.

2.1.1.2. Definición no estricta

Existe un acuerdo general en la comunidad *NER* para considerar como entidades nombradas a otro tipos de entidades que violan el principio de designador rígido. Ejemplos de esto son:

- expresiones temporales como «3 de febrero», «2019».
- expresiones numéricas, como cantidades de dinero y otros tipos de unidades

- expresiones que según contexto denotan una entidad rígida pero no en sí mismas. Por ejemplo «Alfredo Fortabat, empresario argentino, fundador de la compañía cementera Loma Negra.» puede ser considerada una entidad nombrada, sin embargo, el término «Fortabat» por si sólo podría referirse a su viuda «María Amalia Lacroze de Fortabat», al museo de arte «Museo Fortabat» o a la localidad Argentina «Villa Alfredo Fortabat».

En este trabajo hemos priorizado esta definición laxa para tener una mayor expresividad en los tipos de entidades que vamos a detectar. Por lo tanto, en adelante, la definición del término «entidad nombrada» será utilizada bajo una definición no estricta.

2.1.2. Reconocimiento de entidades

El Reconocimiento de entidades nombradas a menudo se divide en dos problemas distintos:

1. Detección de nombres
2. Clasificación de los nombres según el tipo de entidad al que hacen referencia a (persona, organización, ubicación y otro)

En la primera fase los nombres se definen como tramos contiguos de tokens, sin anidamiento, de modo que «Instituto Tecnológico de Buenos Aires» es una entidad única, sin tener en cuenta el hecho de que dentro de esta, la subcadena «Buenos Aires» es en sí otra entidad.

Esta forma de definir el problema, lo reduce a un problema de segmentación.

La segunda fase requiere elegir una ontología para organizar categorías de cosas.

2.1.2.1. Dificultades para encontrar mejores algoritmos

El estado del arte de NER desde 2014 con la introducción de Redes Neuronales ha llegado a una meseta (Honnibal, 2017). En los últimos años el diferencial capitalizado por los diferentes grupos de investigación especializados fue muy reducido. Como puede verse en la siguiente figura:

SYSTEM	TYPE	NER F
spaCy `en_core_web_sm` (2017)	neural	85.67
spaCy `en_core_web_lg` (2017)	neural	86.42
Strubell et al. (2017)	neural	**86.81**
Chiu and Nichols (2016)	neural	86.19
Durrett and Klein (2014)	neural	84.04
Ratinov and Roth (2009)	linear	83.45

Figura 1: Principales avances del estado del arte para NER en los últimos años

Las razones por las cuales esto ocurre escapa el alcance de este trabajo pero se pueden resumir bajo la ley de los rendimientos decrecientes. Se necesita mucho esfuerzo académico para obtener una mejora marginal en el estado del arte actual.

Es por esto que resulta interesante analizar que otro tipo de problemas podemos atacar en este trabajo.

2.1.2.2. Dominios del problema

Existe un hilo conductor en el que todas las investigaciones mencionadas en la figura 1 coinciden; incluso los sistemas NER más avanzadas son frágiles, dado que los sistemas NER desarrollados para un dominio no suelen comportarse bien en otros dominios (Poibeau & Kosseim, 2000). La puesta a punto de un sistema NER para un nuevo dominio conlleva un esfuerzo considerable. Esto es cierto para modelos basados en reglas y para sistemas estadísticos.

Se entiende por dominio a todos los textos que en su conjunto forman un corpus común. Ejemplos de estos son «Noticias periodísticas», «Textos jurídicos», «Reportes militares», «Papers académicos», etc.

2.2. Los datos son el problema

Por todo lo mencionado, es evidente que el cuello de botella para el avance de esta y muchas areas de la IA es el la captura de datos, no los algoritmos (Montani, 2016).

En particular para el estado actual del arte de NER es necesario tener el cuerpo de textos a analizar (Corpus) tagueado de tal manera que se conozcan previamente los nombres y tipos de entidades

de un subconjunto de textos para inferir sobre el resto.

Este aprendizaje se carga en lo que se reconoce como un «Modelo estadístico» y es a ese model al que se le pide inferir nuevos resultados. En nuestra experiencia los modelos pre-entrenados de las diferentes plataformas / librerias / frameworks y trabajos académicos resultan siempre insuficientes para el uso en producción de los mismos.

De esta manera queda bien definido el problema que queremos atacar (y el título de este trabajo)

> Creación eficiente de modelos estadísticos para detección automática y precisa de entidades nombradas

Para este fin se creó un sistema informático que permitirá obtener resultados a la altura de de las soluciones del estado del arte para cualquier corpus de documentos que posea una cantidad necesaria de datos.

3. Estado del arte

El análisis del estado del arte fue basado en la definición del problema.

3.1. Stack de software

Python es el lenguaje más utilizado para resolver problemas de Machine Learning, en especial NLP ("The state of the octoverse," 2019)

Spacy es el framework mejor ranqueado para la tarea de NLP ("The state of the octoverse," 2019) y sabemos por la Figura 1 que obtiene resultados a-la-par del estado del arte actual.

Además la implementación de spacy es robusta y orientada a la creación de apliciones para producción, a diferencia de muchas otras librerías de NLP que sólo se utilizan con fines académicos.

3.2. El pipeline

Todas las operaciones de analisis de lenguaje natural sobre textos no estructurados, tienen como primer paso el de separar el los mismos en tokens. Luego, el documento se procesa en varios pasos diferentes que consisten en el «pipeline de procesamiento». Usualmente los pasos consisten en un etiquetador, un analizador sintáctico y un reconocedor de entidades en el caso de NER.

Cada componente del pipeline devuelve el Doc procesado, que luego se pasa al siguiente componente.

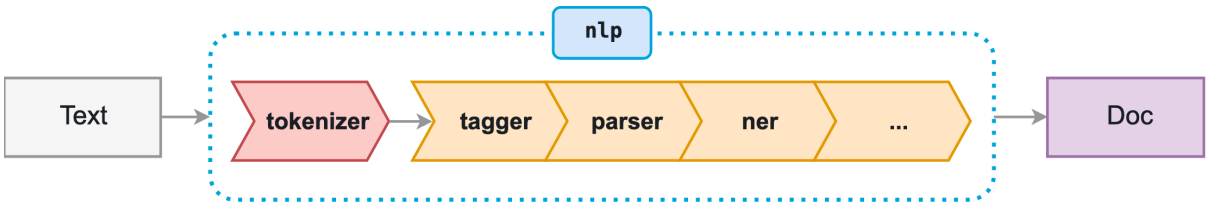


Figura 2: Pipeline standard para los algoritmos de NER

En este capítulo esturiaremos la morfología de dicho pipeline.

3.3. Algoritmo de tokenización

Para tokenizar un texto de manera correcta no basta con separar el mismo en espacios. Dependiendo el lenguaje que se esté estudiando, existen «excepciones» a esta regla y otros caracteres que representan separaciones entre tokens segun el contexto de los mismos.

En particular, spaCy posee algoritmo de tokenización inteligente que puede ser resumido de la siguiente manera:

1. Iterar sobre subcadenas separadas por espacios en blanco.
2. Compruebar si existe una regla definida explícitamente para esta subcadena. Si existe, usarla.

3. De lo contrario, intentar consumir un prefijo. Si consumimos un prefijo, regrese al punto#2, para que los casos especiales siempre tengan prioridad.
4. Si no se puede consumir un prefijo, intente consumir un sufijo y luego regrese al punto #2.
5. Si no se puede consumir un prefijo ni un sufijo, buscar un caso especial.
6. Buscar una coincidencia de token
7. Buscar «infijos» - cosas como guiones, etc. y dividir la subcadena en tokens en todos los infijos.
8. Una vez que no se pueda consumir más de la cadena, tratarla como un token único.

Ejemplo:

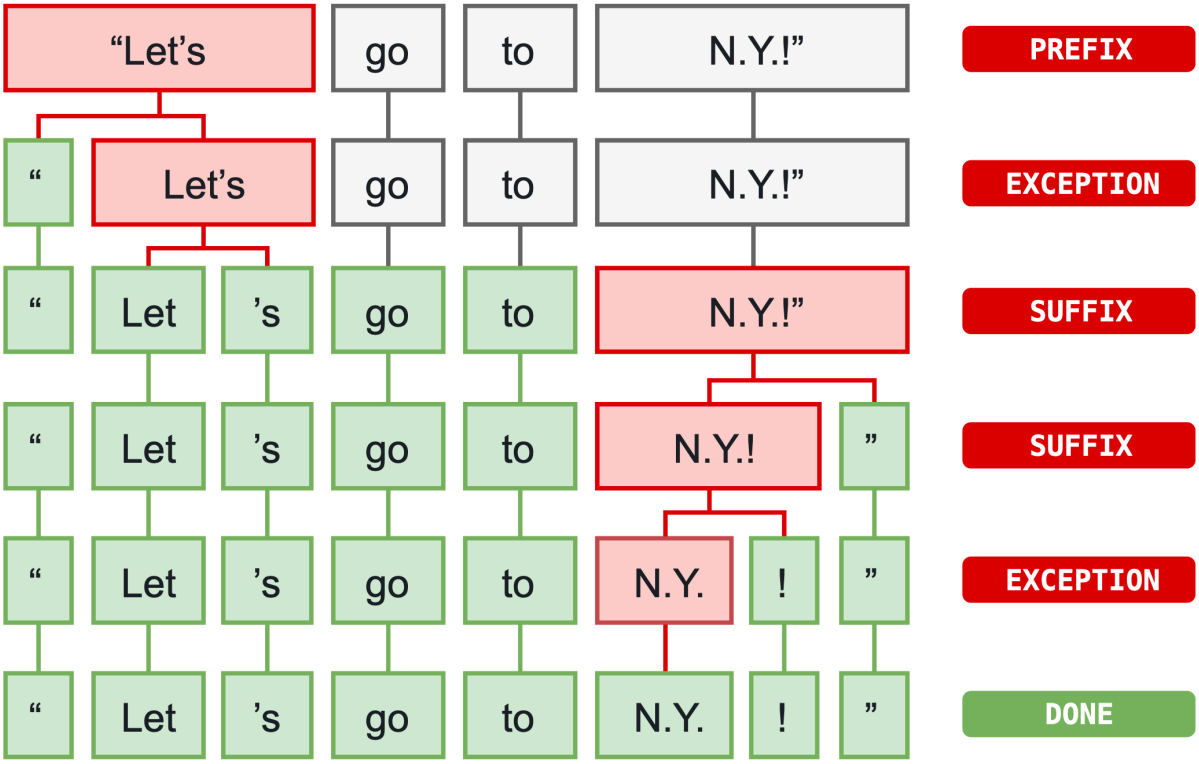


Figura 3: Transiciones del modelo Stack-LSTM indicando la acción aplicada y el estado resultante.

3.4. Reconocimiento de entidades

3.4.1. Modelos basados en reglas

Antes de entrar en detalles de cómo trabaja el modelo estadístico de spacy y entender sus fortalezas es importante esbozar brevemente el grupo de algorítmos más «naive» posible. El de los modelos basados en reglas fijas.

En estos modelos se implementan reglas finitas o expresiones regulares para la detección de las entidades. Las principales limitaciones de este enfoque son:

- **Mucho trabajo manual**: el sistema RB exige un profundo conocimiento del dominio, así como mucho trabajo manual.
- **Consumo de tiempo**: la generación de reglas para un sistema complejo es bastante difícil y requiere mucho tiempo.

- **Menor capacidad de aprendizaje**: el sistema generará el resultado según las reglas, por lo que la capacidad de aprendizaje del sistema por sí mismo es baja.
- **Dominios complejos**: si el corpus demasiado complejo, la creación del sistema RB puede llevar mucho tiempo y análisis. La identificación de patrones complejos es una tarea desafiante en el enfoque RB.

3.4.2. El enfoque de spaCy

Cuando se busca mejorar el aprendizaje automático, generalmente se piensa en la eficiencia y la precisión, pero la dimensión más importante es la generalidad.

La mayoría de los problemas de NLP pueden reducirse a problemas de aprendizaje automático que toman uno o más textos como entrada. Si podemos transformar estos textos en vectores, podemos reutilizar soluciones de aprendizaje profundo (*deep-learning*) de propósito general.

3.4.2.1. Máquina de estados

Experimentos en inglés, holandés, alemán y español muestran que se pueden obtener resultados a-la-par del estado del arte utilizando un autómata finito determinístico de pila en conjunción con una red neuronal (Lample, Ballesteros, Subramanian, Kawakami, & Dyer, 2016)

Este autómata de pila es el nexo entre la Red Neuronal Convolucional (CNN) que contiene el modelo estadístico para predecir entidades y el texto completo. No se envía el texto entero como input a dicha red, sino que se van enviando cada uno de los estados en los que el autómata de pila se mueve para ir generando entidades con una herística del tipo *greedy*.

Las posibles acciones de transición de este autómata son las siguientes:

$\textbf{Out}_t$	$\textbf{Stack}_t$	$\textbf{Buffer}_t$	**Action**	$\textbf{Out}_{t+1}$	$\textbf{Stack}_{t+1}$	$\textbf{Buffer}_{t+1}$	**Segments**
O	S	$(\mathbf{u}, u), B$	SHIFT	O	$(\mathbf{u}, u), S$	B	—
O	$(\mathbf{u}, u), \ldots, (\mathbf{v}, v), S$	B	REDUCE(y)	$g(\mathbf{u}, \ldots, \mathbf{v}, \mathbf{r}_y), O$	S	B	$(u \ldots v, y)$
O	S	$(\mathbf{u}, u), B$	OUT	$g(\mathbf{u}, \mathbf{r}_\varnothing), O$	S	B	—

Figura 4: Transiciones del modelo Stack-LSTM indicando la acción aplicada y el estado resultante.

- SHIFT: consume una token del input y al mueve al stack para generar una nueva entidad.
- REDUCE: mueve el stack actual al output tagueado como entity.
- OUT: consume una token del input y la mueve sl output directamente.

Para saber que acción tomar se consulta el modelo estadístico.En la siguiente figura se puede ver un ejemplo de cómo se recorre una oración bajo el stack propuesto:

Transition	**Output**	**Stack**	**Buffer**	**Segment**
	[]	[]	[Mark, Watney, visited, Mars]	
SHIFT	[]	[Mark]	[Watney, visited, Mars]	
SHIFT	[]	[Mark, Watney]	[visited, Mars]	
REDUCE(PER)	[(Mark Watney)-PER]	[]	[visited, Mars]	(Mark Watney)-PER
OUT	[(Mark Watney)-PER, visited]	[]	[Mars]	
SHIFT	[(Mark Watney)-PER, visited]	[Mars]	[]	
REDUCE(LOC)	[(Mark Watney)-PER, visited, (Mars)-LOC]	[]	[]	(Mars)-LOC

Figura 5: Secuencia de tranciciones para el ejemplo "Mark Watney visited Mars.[en] el modelo de Stack-LSTM.

- Primero se empieza con un stack vacío.
- Se consume «Mark» y la CNN predice que es una posible Persona. Lo envia al stack.
- Se consume «Watney» y la CNN predice que es una posible continuación de Persona. Lo envia al stack.
- Se consume «visited» y la CNN predice que esto no forma parte de una entidad. Por lo tanto antes se REDUCE la entidad «Mark Watney» del stack actual.
- Análogamente se detecta la entidad «Mars»

3.5. El modelo estadístico «deep-learning»

El modelo de deep learning elegido para trabajar es el de spaCy. Consiste en una Red Neuronal convolucional que predice las entidades.

Redes neuronales convolucionales https://es.wikipedia.org/wiki/Redes_neuronales_convolucionales

Para entender cómo funciona dicha red neuronal, se puede definir el proceso en 4 etapas que transforman la información entre diferentes estados

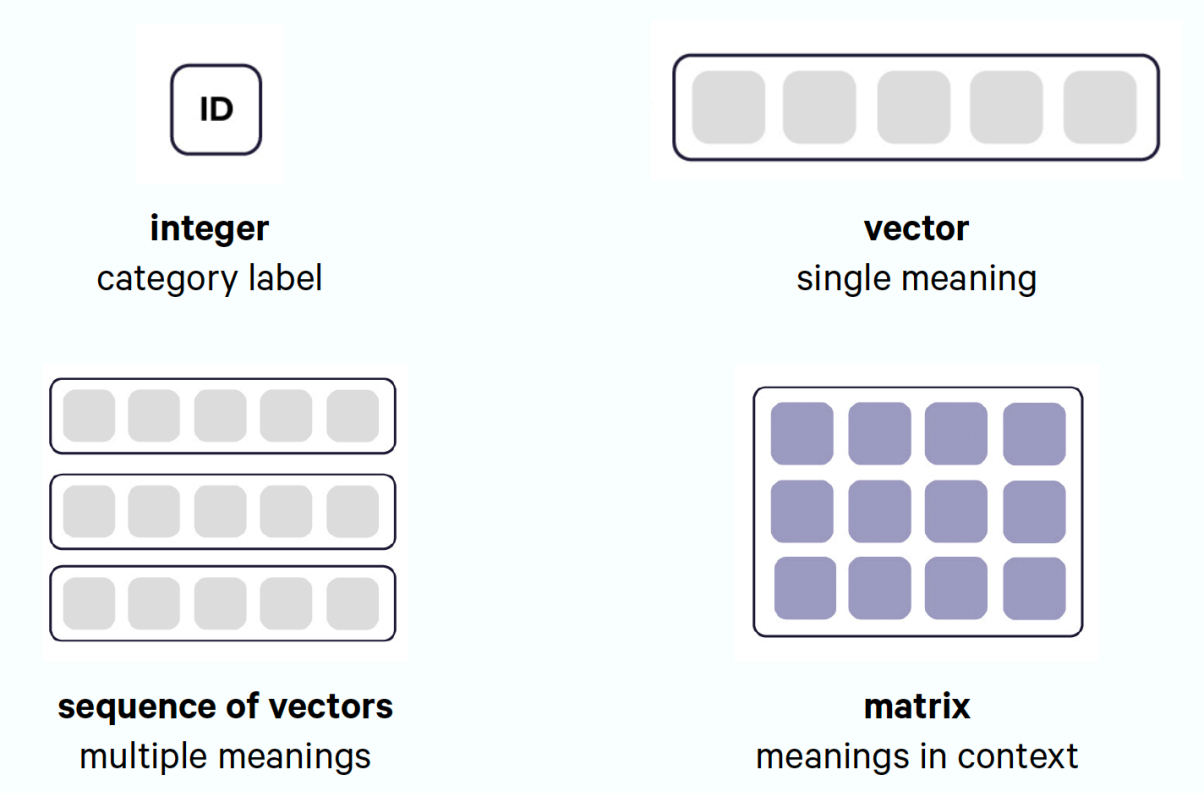


Figura 6: Estados posibles para las diferentes etapas de la CNN

3.5.1. embed

Problema: «todas las palabras sin iguales para la computadora»

La idea de *word embeddings* es la de «embeber» el conjunto de tokens que componen términos con información adicional.

Figura 7: TODO: embed

3.5.2. encode

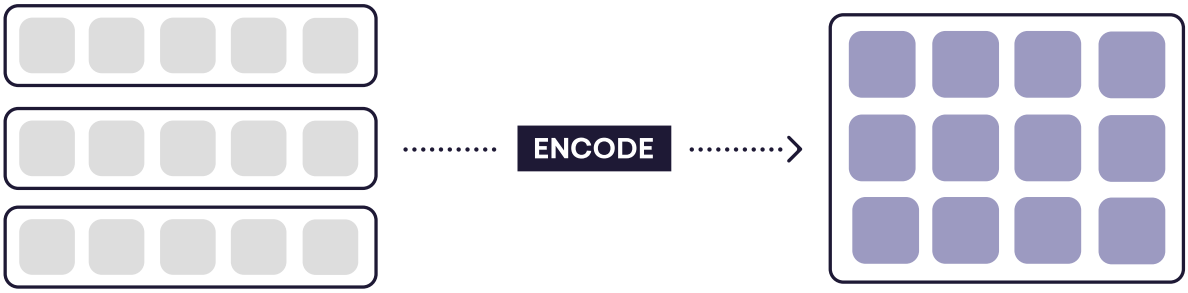


Figura 8: TODO: encode

3.5.3. attend

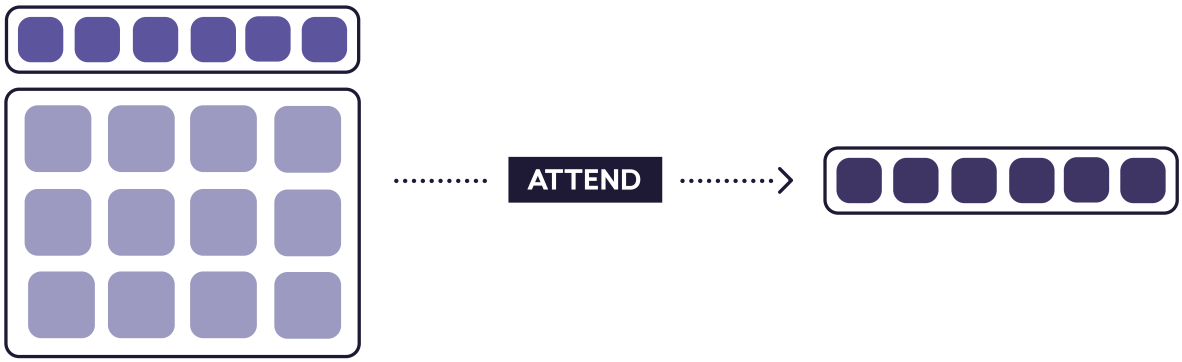


Figura 9: TODO: attend

3.5.4. predict

Figura 10: TODO: predict

Here is a review of existing methods.

3.6. Subword features

Yes, spaCy's NER (and other models) uses subword features, although it doesn't use a character-based CNN to extract them. Instead, the word vectors are learned by concatenating embeddings of NORM, PREFIX, SUFFIX and SHAPE lexical attributes. A hidden layer is then used to allow a non-linear combination of the information in these concatenated vectors. The function for this can be found in spacy._ml.Tok2Vec.

The best reference for this embedding strategy is currently the NER algorithm video: https://www.youtube.com/watch?v=sqDHBH9IjRU

To add to @ honnibal's comment above, there's also a section in the API docs that describes the neural network model architecture in more detail: https://spacy.io/api/#nn-model

3.7. statistical entity recognition model

3.8. Word vectors

$$\vec{king} - \vec{man} + \vec{woman} \approx \vec{queen}$$

(Ethayarajh, Duvenaud, & Hirst, 2019)

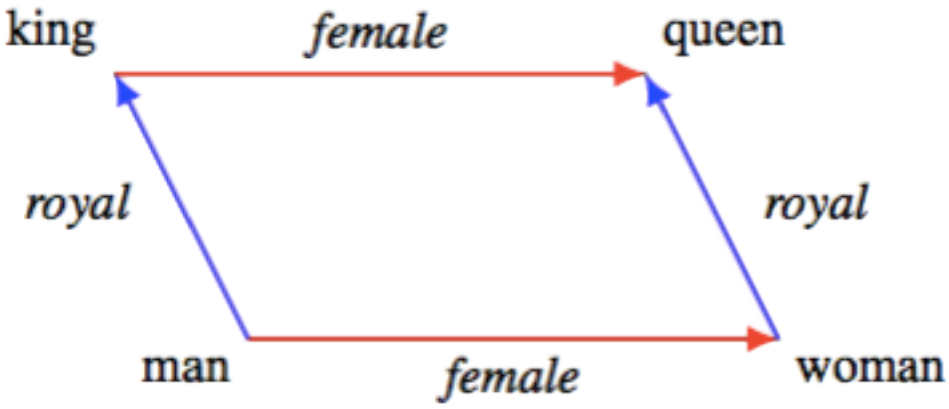


Figura 11: Parallelogram structure in the vector space (by definition)

https://www.youtube.com/watch?v=sqDHBH9IjRU SPACY'S ENTITY RECOGNITION MODEL: incremental parsing with Bloom embeddings & residual CNNs

https://github.com/explosion/talks/blob/master/2018-04-12_Embed-Encode-Attend-Predict.pdf

4. NERd (Implementación)

Definido el problema, queda claro que la creación de un modelo entrenado es de vital importancia para cualquier problema de tagueo de entidades. Es por ello que en el presente proyecto final hemos creado una herramienta para el entrenamiento eficiente de modelos estadísticos así como también una interfaz y API para poder consultar entidades. El nombre de esta herramienta es **NERd**, sigla cuyo significado en inglés es ***N**amed **E**ntity **R**ecognition **D**uh*[1]!

Para organizar este capítulo vamos a realizar una descripción basada en el modelo de vistas de arquitectura 4+1.

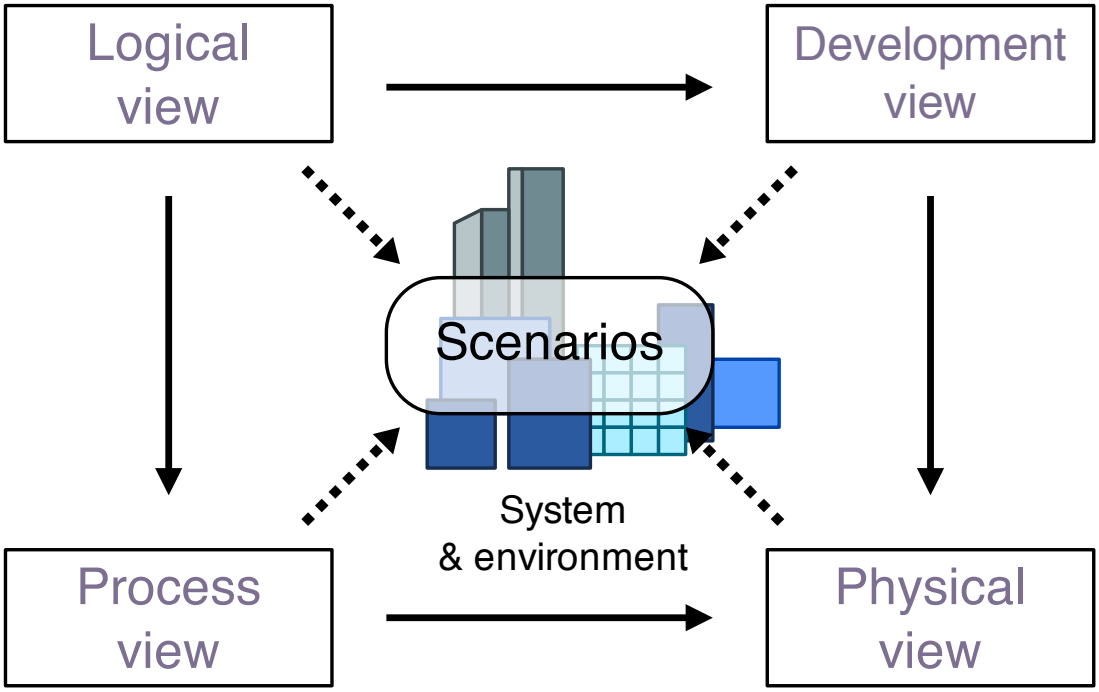


Figura 12: Ilustración de arquitectura 4+1

Este modelo nos permite describir la aplicación de una manera genérica y ordenada.

> *The «4+1» view model is rather «generic»: other notations and tools can be used, other design methods can be used, especially for the logical and process decompositions, but we have indicated the ones we have used with success.*
>
> — (Kruchten, 1995)

4.1. Vista lógica

> La vista lógica se refiere a la funcionalidad que el sistema proporciona a los usuarios finales.

[1] Expresión de obviedad. *Used to express your belief that what was said was extremely obvious* ("Duh definition," 2019)

A continuación detallamos distintas partes del servicio NERd así como también de la interfaz de entrenamiento.

4.1.1. Servicio

El acceso al servicio se realiza mediante un API REST que se auto-documenta debido a implementar la especificación de OpenAPI. A continuación detallamos los endpoints.

4.1.1.1. Autenticación

Rutas del API dedicadas a la autenticación de usuarios.

- **POST** */api/auth/register*
  - Registrar un usuario nuevo.
- **POST** */api/auth/token*
  - Genera un nuevo token de acceso y refresco con credenciales.
  - Utilizado para la funcionalidad de *login*
- **POST** */api/auth/refresh*
  - Refresca el token de acceso.
  - Utilizado cuando un token de acceso caducó.

4.1.1.2. Usuarios

Conjunto de operaciones relacionadas con los usuarios del sistema.

- **GET** */api/users*
  - Lista de usuarios existentes
  - Separa los resultados en páginas
- **GET** */api/users/top5*
  - Lista de los 5 usuarios con más entrenamientos
- **GET** */api/users/me*
  - Retorna la información del usuario logueado
- **PATCH** */api/users/me*
  - Actualiza la información del usuario logueado
- **GET** */api/users/me/trainings*
  - Retorna los entrenamientos del usuario logueado
  - Separa los resultados en páginas
- **GET** */api/users/{user_id}*
  - Retorna la información del usuario especificado por *user_id*
- **PATCH** */api/users/{user_id}*
  - Actualiza la información del usuario especificado por *user_id*
- **DELETE** */api/users/{user_id}*
  - Borra al usuario especificado por *user_id*
- **GET** _/api/users/{user_id}/trainings
  - Retorna los entrenamientos del usuario especificado

4.1.1.3. Roles

- **GET** */api/roles*
  - Retorna la lista de todos los roles asignables a usuarios del sistema

4.1.1.4. Corpus

Rutas dedicadas a operaciones con el corpus del sistema.

- **GET** */api/corpus/{text_id}*
  - Retorna los detalles del texto especificado por *text_id*
- **DELETE** */api/corpus/{text_id}*
  - Borra un texto especificado por *text_id* del corpus
- **GET** */api/corpus/{text_id}/trainings*
  - Retorna la lista de entrenamientos proporcionados por los usuarios sobre las entidades en el texto
- **PUT** */api/corpus/{text_id}/trainings*
  - Agrega un entrenamiento para el texto con id *text_id*
- **POST** */api/corpus/upload*
  - Permite agregar textos de manera masiva al sistema
  - Acepta una lista de archivos .txt donde cada línea es un texto a agregar
  - Los archivos deben ser UTF-8
- **GET** */api/corpus*
  - Lista de textos cargados en el sistema para entrenamiento
  - Separa los resultados en páginas
- **POST** */api/corpus*
  - Agrega un texto al sistema para entrenamiento

4.1.1.5. Snapshots

Conjunto de operaciones relacionadas con los snapshots y workers.

- **GET** */api/snapshots*
  - Listado de los snapshots disponibles
  - Separa los resultados en páginas
- **GET** */api/snapshots/{snapshot_id}*
  - Retorna información (tipos de entidades, fecha de creación, fecha de entrenamiento, etc.) sobre un snapshot específico
- **DELETE** */api/snapshots/{snapshot_id}*
  - Borra un snapshot con el id especificado
- **POST** */api/snapshots/{snapshot_id}/force-train*
  - Envía la tarea de entrenamiento a los workers que tienen el snapshot *snapshot_id* cargado.
- **POST** */api/snapshots/{snapshot_id}/force-untrain*
  - Envía la tarea de desentrenar a los workers que tienen el snapshot *snapshot_id* cargado.
- **GET** */api/snapshots/current*
  - Retorna información sobre el snapshot actual
- **PUT** */api/snapshots/current*
  - Crea un nuevo snapshot con la información provista

4.1.1.6. Reconocimiento de Entidades Nombradas

Conjunto de operaciones relacionadas al *Reconocimiento de Entidades Nombradas*

- **GET** */api/ner/train*
  - Retorna un texto para que un usuario del sistema revise si está correctamente inferido
  - Únicamente retorna textos que el usuario logueado no haya corregido ya
- **GET** */api/ner/compare/{first_snapshot}/{second_snapshot}*
  - Compara el *Reconocimiento de Entidades Nombradas* entre dos snapshots distintos
- **POST** */api/ner/current/parse*
  - Retorna un documento Spacy para un texto dado utilizando el snapshot actual
- **POST** */api/ner/{snapshot_id}/parse*
  - Retorna un documento Spacy para un texto dado utilizando el snapshot especificado
- **POST** */api/ner/current/entities*
  - Retorna la lista de *Entidades Nombradas* para un texto dado utilizando el modelo actual
- **POST** */api/ner/{snapshot_id}/entities*
  - Retorna la lista de *Entidades Nombradas* para un texto dado utilizando el modelo especificado

4.1.1.7. Entrenamientos

- **DELETE** */api/trainings/{training_id}*
  - Borra un entrenamiento

4.1.1.8. Workers

- **GET** */api/workers/*
  - Lista de los workers disponibles
- **POST** */api/workers/reassign*
  - Reasigna un trabajador de un a versión de snapshot a otra

4.1.2. Web

La página web de *NERd* está enfocada en las tareas de mantenimiento de los servicios ofrecidos por el *API* así como también ofrece de interfaces que permiten a usuarios del sistema corregir de manera eficiente el modelo de inferencia.

4.1.2.1. Inicio

Pantalla de inicio donde se encuentran accesos rápidos para entrenar el modelo o para poder buscar entidades en textos. También se encuentra aquí una lista de los 5 usuarios que más contribuyeron a entrenar el modelo. Detrás de esta funcionalidad se busca generar un espíritu competitivo entre los usuarios para que los mismos busquen contribuir más.

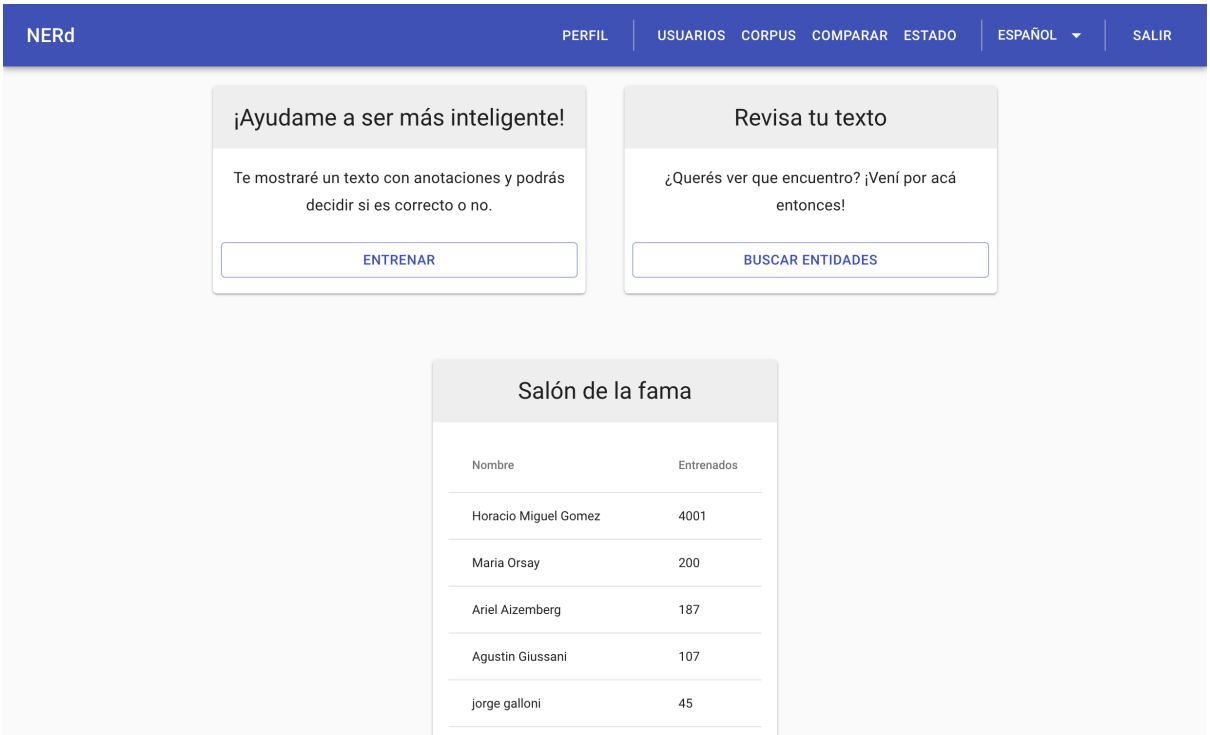


Figura 13: Pantalla de inicio con usuario logueado

Si la persona no cuenta con permisos de entrenador, se le sugiere que contacte a un administrador para que le otorgue el permiso.

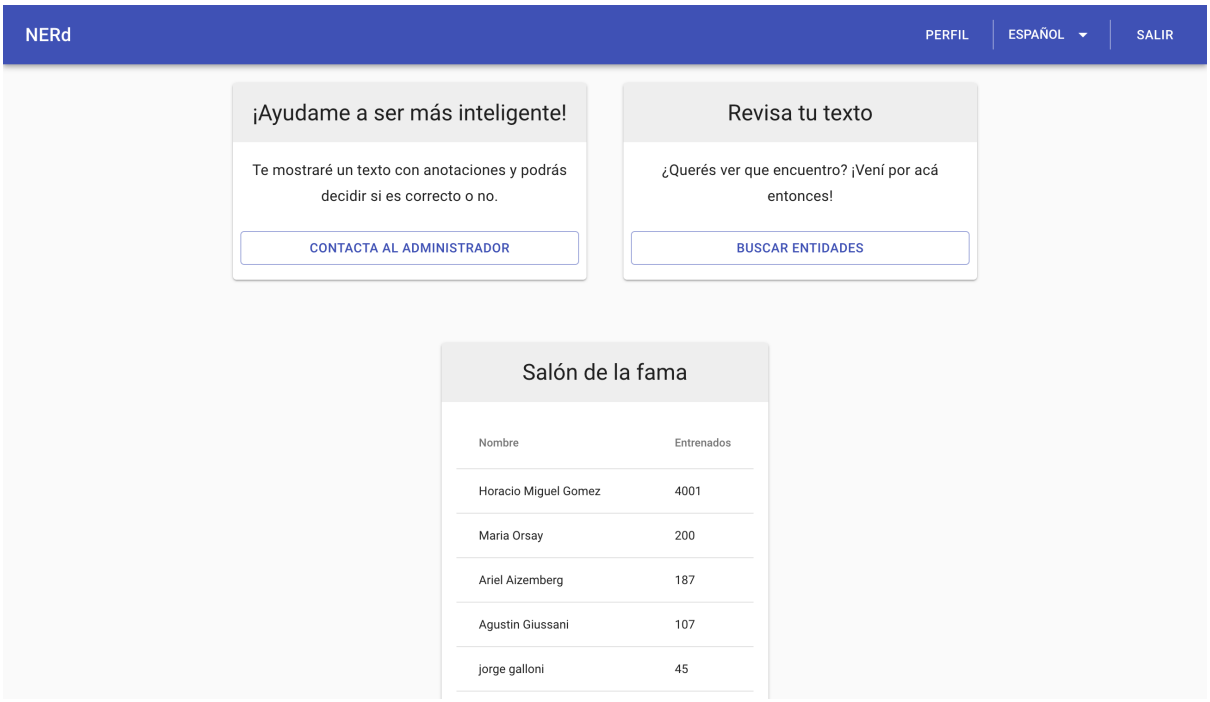


Figura 14: Pantalla de inicio sin rol de entrenador

Si la persona visitando la página no cuenta con una sesión activa, se le invita a ingresar con una cuenta pre-existente o a registrarse.

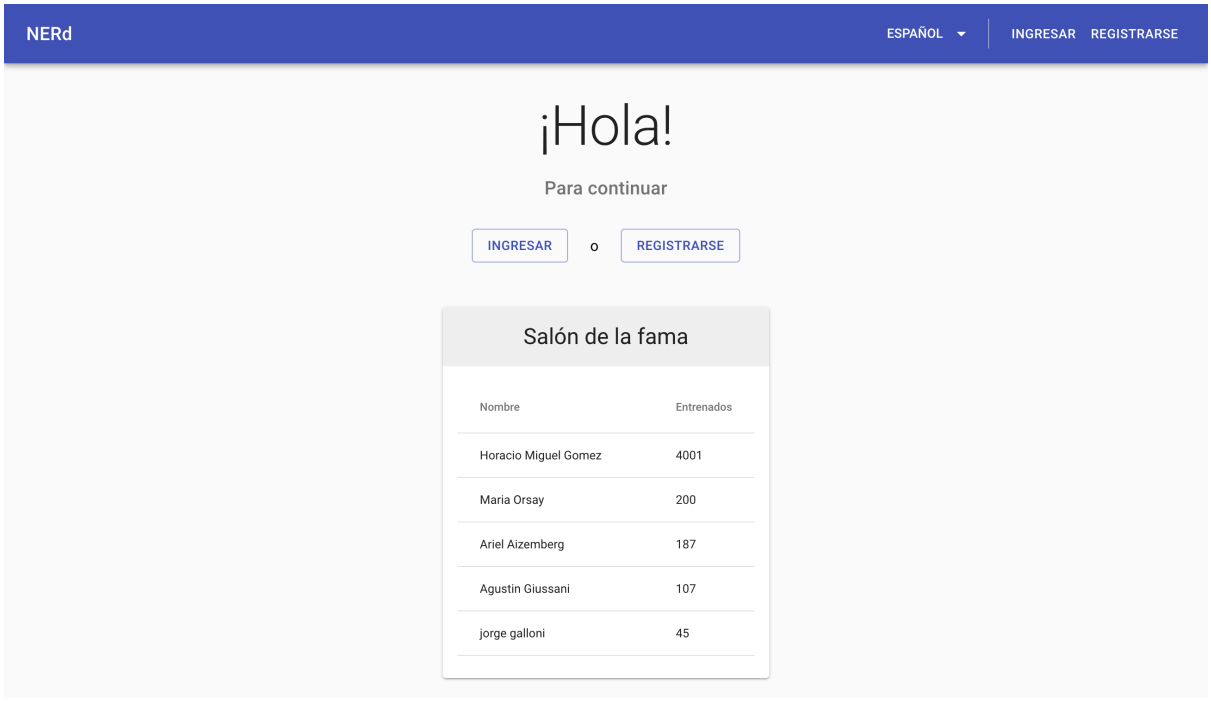


Figura 15: Pantalla de inicio sin sesión

4.1.2.2. Entrenamiento

La pantalla de *Entrenamiento* es el núcleo de la web en la cual es posible entrenar el modelo.

El usuario es presentado con un texto perteneciente al Corpus del servicio con las entidades inferidas por el modelo actual. Con un editor especial, le permitimos al usuario poder corregir las entidades inferidas y enviarle la corrección al servicio. Esa corrección será utilizada posteriormente a la hora de mejorar el modelo actual.

Figura 16: Pantalla de entrenamiento

4.1.2.2.1. Usabilidad

Tuvimos un foco fuerte en la usabilidad del widget ya que los entrenadores del servicio van a pasar prácticamente todo su tiempo en ésta pantalla, por lo que se tuvieron las siguientes consideraciones en la implementación.

Llamado a acción y ayuda

Dado que lo primero que ve el usuario es un texto con anotaciones, agregamos un título que invita al usuario a realizar acciones sobre el texto. De esta manera, le mostramos las dos acciones principales realizables desde el widget de entrenamiento: Click en alguna palabra o entidad y arrastrar un conjunto de palabras para crear una entidad nueva. Como refuerzo de este llamado a acción, agregamos un botón que al ser clickeado muestra un mensaje de ayuda con instrucciones más detalladas sobre el objetivo del entrenador y las acciones que deben de realizarse sobre el mismo.

Figura 17: Ayuda del entrenador

Creación y edición de entidades

Para la creación de entidades decidimos ofrecer dos maneras: La primera es arrastrando un conjunto de palabras de manera tal de unirlas todas en una única entidad. La otra es hacer click en una palabra y ahí se ofrecen opciones dependiendo de la ubicación de la palabra dentro del texto:

- Si no existen entidades en el texto actual, se le asigna por defecto el tipo *MISC* y se muestran el resto de los tipos para permitir cambiarlo de ser necesario.
- Si existen entidades antes o después, se ofrece la opción de unir la palabra actual con la entidada más próxima para el lado elegido.

Figura 18: Edición de entidad

Para la edición de entidades decidimos permitir únicamente la modificación del tipo de una entidad inferida. Si el modelo infirió una entidad de manera incorrecta, ya sea por que sea una entidad inválida o agregó palabras de más a una entidad inválida, permitimos que el usuario remueva la entidad y que después vuelva a agregar la entidad correcta.

Optimización en tiempos de carga

Dado que es esperado que un usuario entrene más de un texto, al momento de pedir un texto para mostrar, se pide el siguiente. Mediante este mecanismo de pre-carga, podemos eliminar el tiempo de espera entre texto y texto ofreciendo al usuario una experiencia completamente fluida.

4.1.2.3. Administración de usuarios

La pantalla de *Administración de usuarios* permite a los usuarios con el rol de administrador poder modificar los roles de todos los usuarios del sistema, borrarlos o acceder a los detalles del usuario, tal como la lista de textos entrenados.

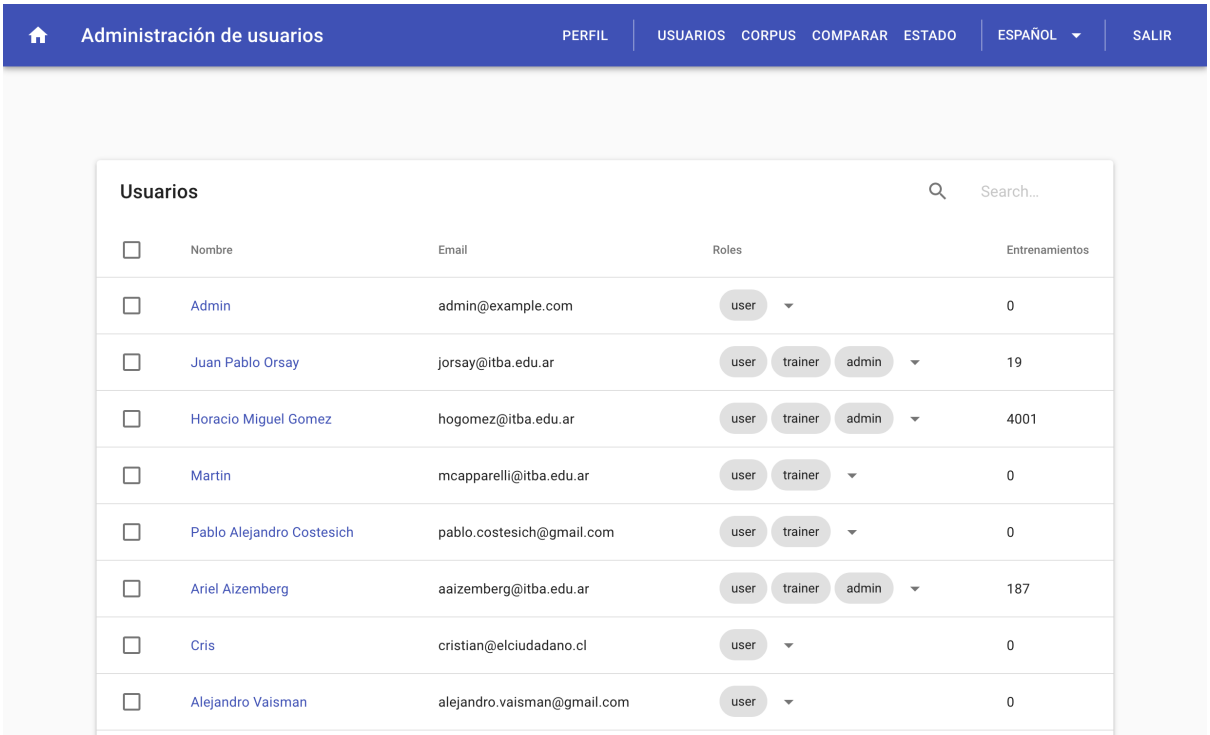


Figura 19: Administración de usuarios

4.1.2.4. Detalles de usuario

La pantalla de detalle de usuario permite al usuario con sesión activa ver sus entrenamientos y cambiar su contraseña.

Los usuarios con rol administrador pueden realizar las acciones mencionadas previamente pero a otros usuarios.

Figura 20: Perfil de usuario

4.1.2.5. Corpus

La pantalla de *Corpus* permite a un usuario con el rol de administrador realizar tareas relacionadas con el corpus del sistema.

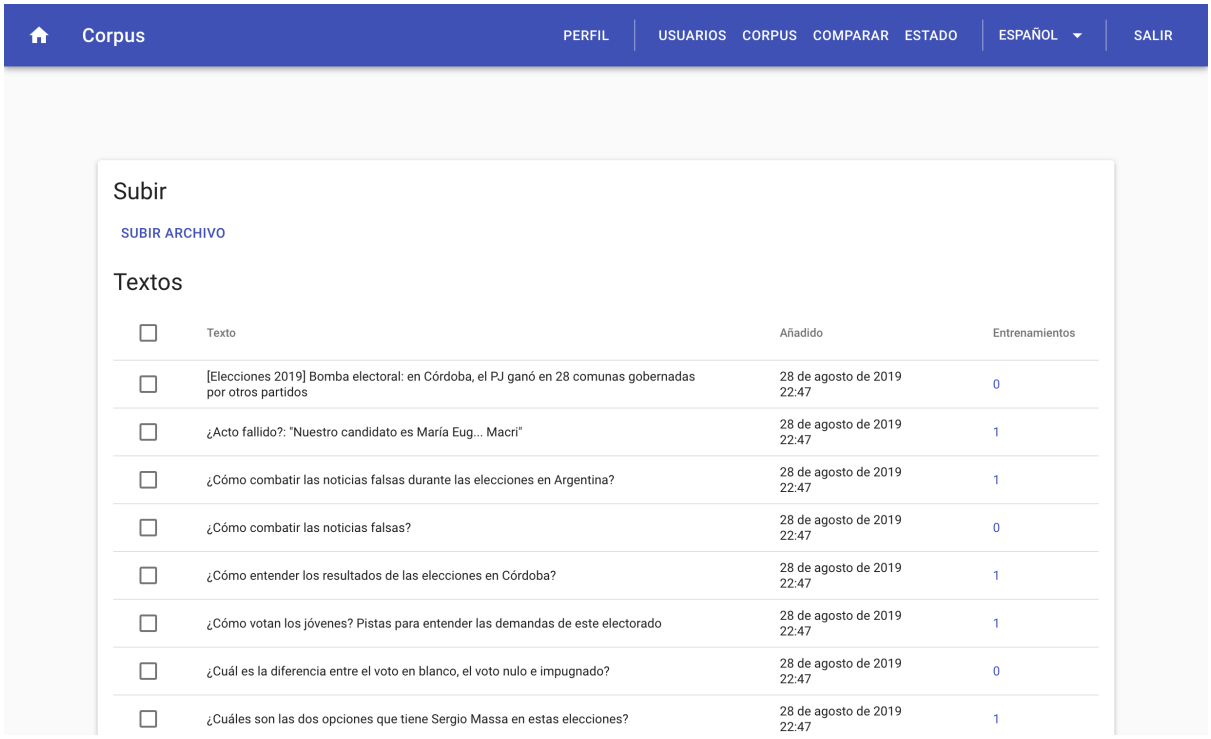


Figura 21: Administración de corpus

Desde aquí es posible agregar textos al corpus utilizando la funcionalidad de subida de archivos. Los archivos deben ser archivos con extensión *.txt* y cada línea del archivo será agregada al corpus como un texto individual.

También es posible desde aquí ver todos los textos que forman parte del corpus así como también poder ver los entrenamientos para cada uno de los textos. Finalmente, es posible quitar textos del corpus así como también es posible eliminar correcciones a las inferencias de entidades cargados por usuarios.

4.1.2.6. Estado

La pantalla de *Estado* permite a un usuario con el rol de administrador visualizar el estado de entrenamiento del corpus así como también realizar diversas acciones sobre los *workers*.

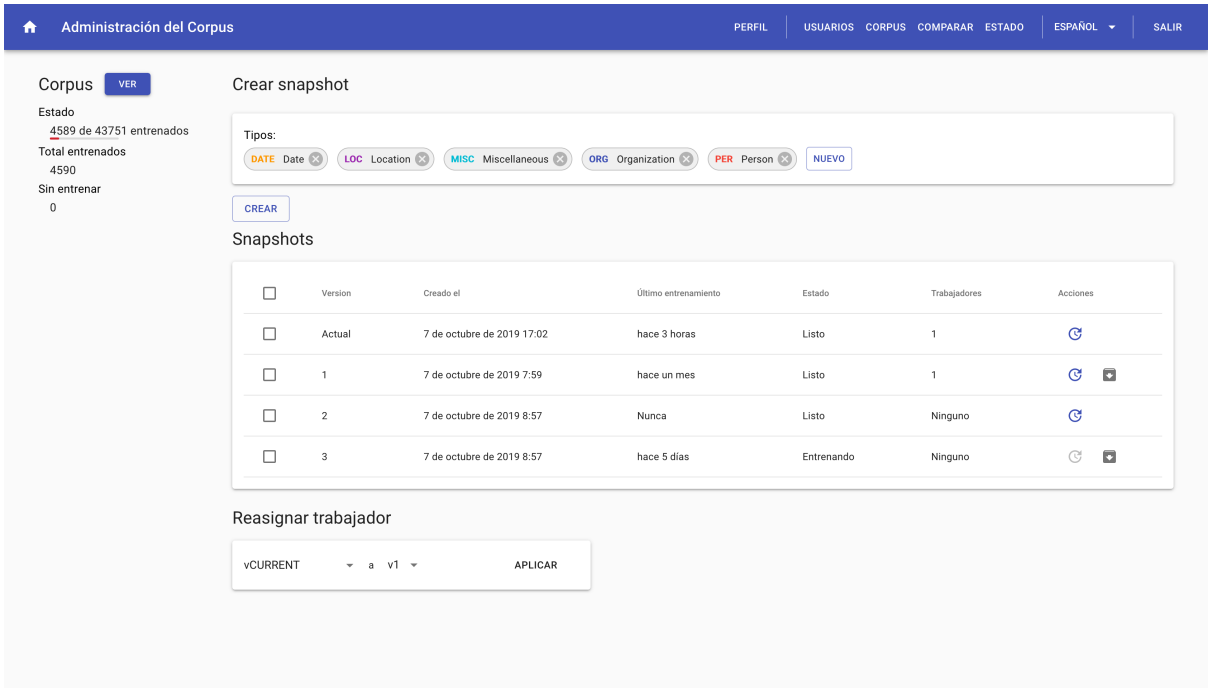


Figura 22: Información de corpus y manejo de workers

4.1.2.6.1. Secciones

Corpus

Es la columna la izquierda y aquí se puede ver rápidamente que porcentaje de el corpus contiene correcciones por usuarios así como también saber la cantidad total de correcciones del sistema (un texto puede tener más de una corrección por distintos usuarios) y también presenta un botón que permite al administrador ir a la pantalla de *Corpus*.

Crear snapshot

Es la sección en la cual será posible crear, borrar o modificar los tipos de entidades reconocidos por el snapshot actual. La acción de editar las entidades genera un snapshot nuevo.

Si el administrador así lo quisiera, puede utilizar esta sección para crear un snapshot nuevo sin editar entidades.

Snapshots

Sección en la cual podemos ver la lista completa de snapshots. Para cada Snapshot, se muestra cuando fue la última vez que se entrenó así como también cuantos trabajadores tiene asignados. Finalmente es posible desde aquí forzar a entrenar el modelo para ese snapshot en particular y también se presenta la opción para desentrenar, borrando el modelo guardado en el disco.

Reasignar trabajador

Sección que permite reasignar trabajadores para que sirvan un snapshot distinto. De esta manera se pueden servir distintas versiones del modelo de inferencia para poder realizar distintas pruebas sobre los mismos.

4.1.2.7. Sandbox

La pantalla de *Sandbox* permite a los usuarios hacer consultas al servicio NERd para poder obtener entidades nombradas a partir de textos arbitrarios. Adicionalmente, si el usuario tiene el rol de entrenador, podrá corregir las entidades inferidas y agregar el texto con sus correcciones al corpus.

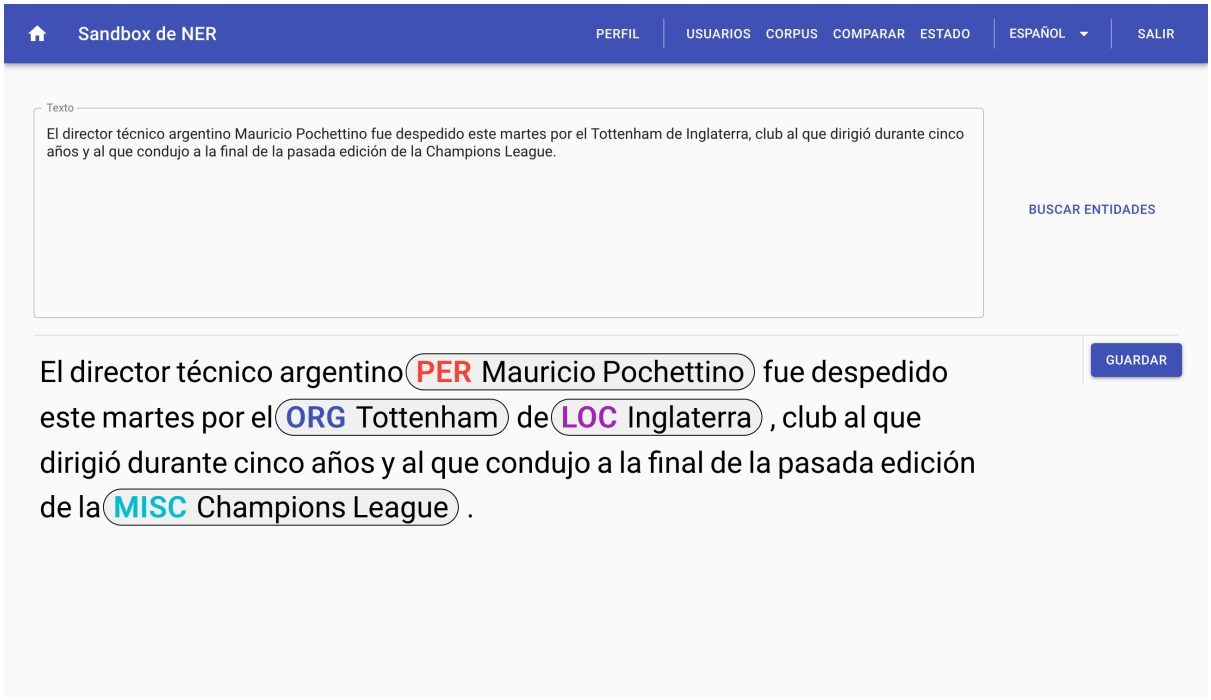


Figura 23: Inferencia de entidades en sandbox

4.1.2.8. Comparar

Sección accesible únicamente a administradores en la que es posible comparar las entidades inferidas por dos modelos distintos. A su vez, si el usuario logueado tiene el permiso de entrenador, es posible corregir de manera inline los errores en la inferencia del modelo actual.

Figura 24: Comparativa de modelos

4.2. Vista de proceso

> La vista de proceso trata los aspectos dinámicos del sistema, explica los procesos del sistema y cómo se comunican, y se centra en el comportamiento del sistema en tiempo de ejecución. La vista de proceso aborda concurrencia, distribución, integradores, rendimiento y escalabilidad, etc.

Existen cinco procesos que juntos forman la totalidad de NERd y su entrenador.

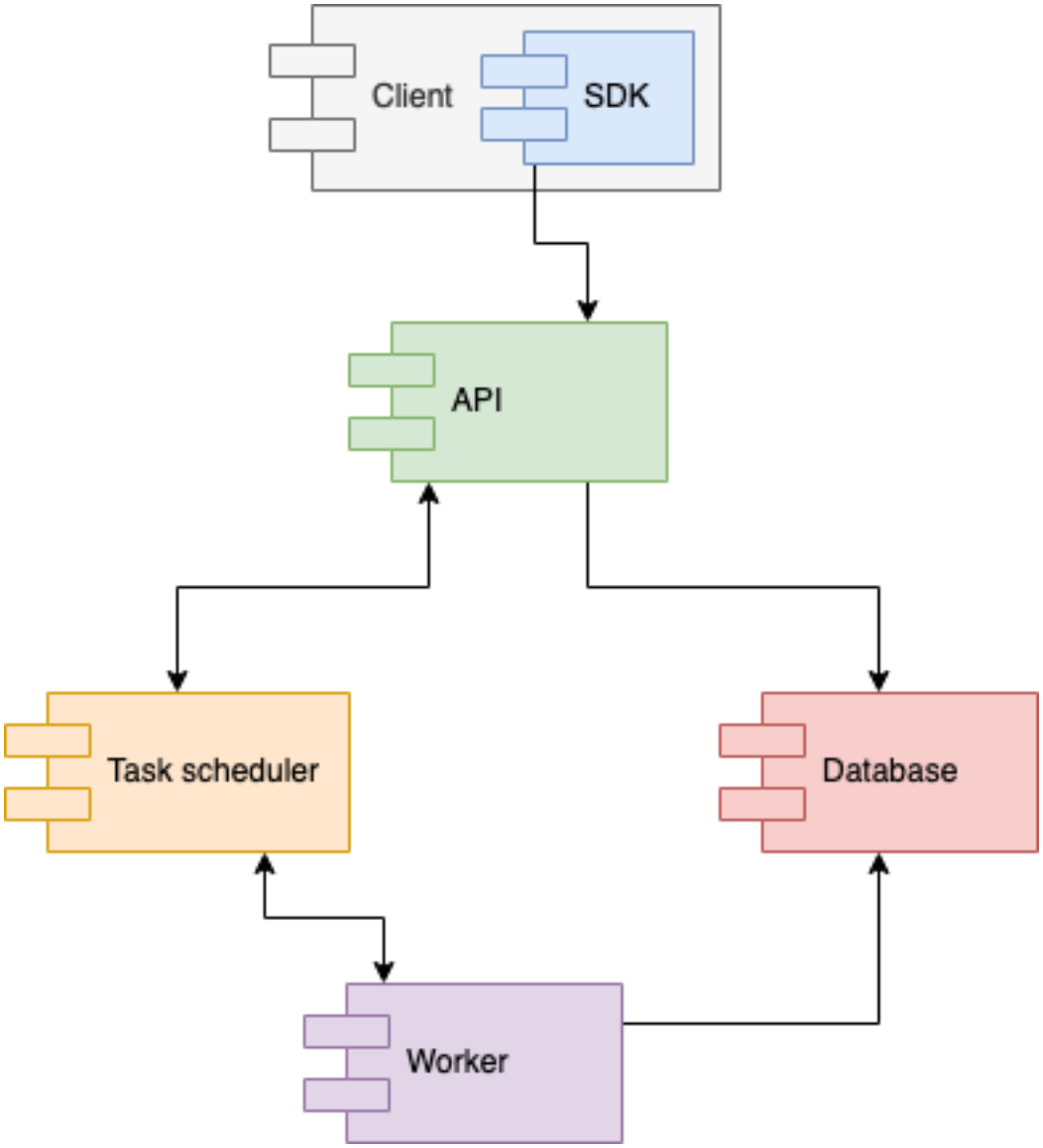


Figura 25: Componentes del sistema

4.2.1. API

Servicio en el cual los clientes pueden pedir las *Entidades Nombradas* de textos así como también permite la carga de textos con sus respectivas correcciones que luego serán utilizadas para entrenar el modelo de inferencia. El *API* se comunica el servicio de *Bases de Datos* para obtener o modificar información sobre los usuarios, los textos pertenecientes al corpus o información de los distintos *snapshots* y con el *Task Scheduler* para realizar diversas tareas relacionadas con *Spacy*.

4.2.2. Task Scheduler

Recibe tareas desde el *API* tales como reconocer entidades, entrenar y desentrenar un modelo, cambiar el modelo de un worker entre otras. Dado que su tarea es recibir tareas desde el *API* y enviar y recibir mensajes de los *Workers*, debería existir una única instancia del mismo. A su vez, notifica al *API* la finalización de tareas asincrónicas

4.2.3. Worker

Un *Worker* se encarga de realizar el *Reconocimiento de Entidades Nombradas* así como también de entrenar modelos de inferencia. Es un servicio que existe de manera independiente y necesita de dos otros servicios para funcionar:

- *Base de datos* para obtener los datos necesarios para entrenar un modelo.
- *Task scheduler* para poder recibir tareas.

Como el *Worker* es una unidad de trabajo que recibe tareas desde el *Task Scheduler*, es posible tener varios *Workers* donde cada uno utiliza un modelo de inferencia distinto (generados a partir de distintos snapshots).

4.2.4. Database

Servicio que contiene la base de datos con la información necesaria (usuarios, snapshots) para que el servicio de API funcione así como también los datos necesarios por los workers (entrenamientos).

4.2.5. Cliente

Es la interfaz del entrenador que se comunica con el API utilizando un SDK auto-generado a partir de la definición de *OpenAPI*. Implementado en este proyecto es un cliente web el cual permite explotar las funcionalidades de entrenamiento así como de inferencia de entidades

4.3. Vista de desarrollo

> La vista de desarrollo ilustra un sistema desde la perspectiva de un programador y se ocupa de la gestión de software. Esta vista también se conoce como la vista de implementación.

4.3.1. Web

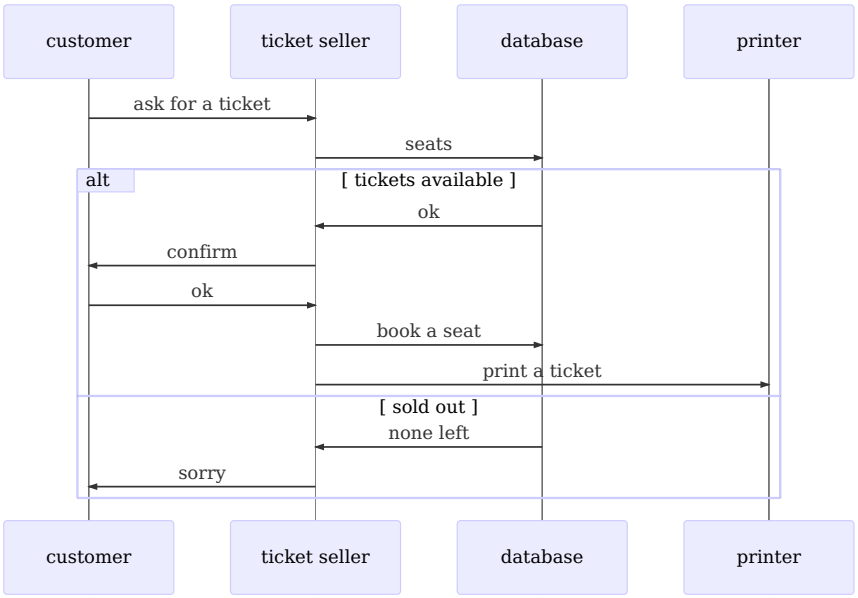
La página web que contiene la interfaz de administración así como también la de entrenamiento fue implementada utilizando el lenguaje *Typescript*

4.4. Vista física

> La vista física representa el sistema desde el punto de vista de un ingeniero de sistemas. Se refiere a la topología de los componentes de software en la capa física, así como a las conexiones físicas entre estos componentes. Esta vista también se conoce como la vista de *deployment*.

4.5. Escenarios

La descripción de una arquitectura se ilustra utilizando un pequeño conjunto de casos de uso, o escenarios, que se convierten en una quinta vista. Los escenarios describen secuencias de interacciones entre objetos y entre procesos. Se utilizan para identificar elementos arquitectónicos y para ilustrar y validar el diseño de la arquitectura. También sirven como punto de partida para las pruebas de un prototipo de arquitectura. Esta vista también se conoce como vista de caso de uso.



5. Resultados

Matriz de confusión -> dos indicadores! true possitive vs true negative Credibility: evaluation whats been leaerned.

5.1. Métrica: precisión y exhaustividad

("Precision and recall," 2019)

> TODO: reescribir
>
> En conferencias académicas como CoNLL, se ha definido una variante del [[valor-F]] de la siguiente manera:
>
> - Precisión es el número de entidades nombradas que coinciden exactamente con conjunto de evaluación. I.e. cuando se predice [Persona Hans] [Persona Blick], pero lo correcto era [Person Hans Blick], la precisión es cero. La precisión es después promediada por cada una de la entidades nombradas.
> - El recobrado es el número de entidades del conjunto de evaluación que aparecen exactamente en la misma posición en las predicciones.
> - El Valor-F es la media armónica de estos dos valores. Se deriva de la anterior definición que cualquier predicción que reconozca erróneamente un token como parte de una entidad nombrada o que deje de detectar un token que sí es una entidad nombrada o lo clasifique erróneamente no contribuirá ni a la precisión ni al recobrado.

=== Evaluación formal === Para evaluar la calidad de la salida de un sistema NER, se han definido varias medidas. Las medidas habituales se llaman [[Precision_and_recall | Precisión, recuperación]] y [[Puntuación F1]]. Sin embargo, quedan varios problemas sobre cómo calcular esos valores.

Estas medidas estadísticas funcionan razonablemente bien para los casos obvios de encontrar o faltar una entidad real exactamente; y para encontrar una no entidad. Sin embargo, NER puede fallar de muchas otras maneras, muchas de las cuales son posiblemente «parcialmente correctas», y no deben considerarse como éxitos o fracasos competitivos. Por ejemplo, identificar una entidad real, pero: * con menos tokens de los deseados (por ejemplo, falta el último token de «John Smith, M.D.») * con más tokens de los deseados (por ejemplo, incluyendo la primera palabra de «The University of MD») * particionando entidades adyacentes de manera diferente (por ejemplo, tratando a «Smith, Jones Robinson» como entidades 2 vs. 3) * asignarle un tipo completamente incorrecto (por ejemplo, llamar a un nombre personal a una organización) * asignándole un tipo relacionado pero inexacto (por ejemplo, «sustancia» vs. «droga», o «escuela» vs. «organización») * identificar correctamente una entidad, cuando lo que el usuario quería era una entidad de menor o mayor alcance (por ejemplo, identificar «James Madison» como un nombre personal, cuando forma parte de la «Universidad James Madison». Algunos sistemas NER imponen la restricción que las entidades nunca se superpongan o aniden, lo que significa que en algunos casos uno debe tomar decisiones arbitrarias o específicas de la tarea.

Un método demasiado simple para medir la precisión es simplemente contar qué fracción de todas las fichas en el texto se identificaron correcta o incorrectamente como parte de referencias de entidad (o como entidades del tipo correcto). Esto tiene al menos dos problemas: en primer lugar,

la gran mayoría de los tokens en el texto del mundo real no forman parte de los nombres de las entidades, por lo que la precisión de la línea de base (siempre predice «no una entidad») es extravagantemente alta, típicamente> 90 %; y segundo, predecir erróneamente el lapso completo del nombre de una entidad no se penaliza adecuadamente (encontrar solo el nombre de una persona cuando le sigue su apellido podría calificarse como ½ precisión).

En conferencias académicas como CoNLL, una variante de la [[puntuación F1]] se ha definido de la siguiente manera: {{r | conll03intro}}

- [[Precisión y recuperación | Precisión]] es el número de intervalos de nombre de entidad pronosticados que se alinean " exactamente " con intervalos en los datos de evaluación [[Verdad fundamental # Estadísticas y aprendizaje automático | estándar de oro]]. Es decir. cuando se predice [Persona Hans] [Persona Blick] pero se requiere [Persona Hans Blick], la precisión del nombre predicho es cero. La precisión se promedia sobre todos los nombres de entidad pronosticados.
- Recordar es igualmente el número de nombres en el estándar de oro que aparecen exactamente en la misma ubicación en las predicciones.
- La puntuación F1 es la [[media armónica]] de estos dos.

De la definición anterior se deduce que cualquier predicción que omita un solo token, incluye un token espurio o tiene clase incorrecta, es un error difícil y no contribuye positivamente ni a la precisión ni a la recuperación. Por lo tanto, se puede decir que esta medida es pesimista: puede darse el caso de que muchos «errores» estén cerca de ser correctos, y podrían ser adecuados para un propósito dado. Por ejemplo, un sistema siempre puede omitir títulos como «Sra.» o «Ph.D.», pero se compara con un sistema o datos de verdad que esperan que se incluyan títulos. En esos casos, cada nombre se trata como un error. Debido a tales problemas, es importante examinar los tipos de errores y decidir qué tan importantes se les dan los objetivos y requisitos.

6. Discusión

6.1. Tipos de entidades relevantes

tener en cuenta (Brunstein, 2002)

```
# Notas sobre mejora en tipos de entidades
Presidente -> Person Descriptor
NORP -> (Polical) Peronistas, Kirchneristas
Facility Name -> usually location. "Wall Street", "Muralla China"

Organization Name -> Government vs Corporation.
Product Name -> autos "Fiat Toro", celulares "Galaxy S10"
Events -> Superclásico. Superliga. Copa argentina. Elecciones 2019. Las Paso.
Disease ->
Game -> Football, Basket (para "titulos" no tan relevante)
```

6.2. Seed en los types

en especial para los nuevos.

6.3. Linkeo de entidades con Knowledge Base

La tarea de reconocer entidades nombradas en el texto es Reconocimiento de entidades nombradas, mientras que la tarea de determinar la identidad de las entidades nombradas mencionadas en el texto se llama Desambiguación de entidades nombradas. Ambas tareas requieren algoritmos y recursos dedicados para ser abordados. [3]

6.4. Mejora live vs offline

Mejora «Uncertainty sampling» -> buscar entidades que tengan un score ~ 0.5

6.5. Utilidad de la herramienta

Para poder poner a prueba nuestra herramienta **NERd** en un entorno real participamos de la hackaton en MediaParty 2019.

> **("Hackaton," 2019)** es un evento de tres días en Argentina, que reúne a 2500 emprendedores, periodistas, programadores de software y diseñadores de cinco continentes para trabajar juntos para el futuro de los medios de comunicación. Nacido de Hacks/Hackers Buenos Aires, el evento fusiona a grandes empresas como New York Times, The Guardian, Vox, ProPublica, Watchup, Neo4J o DocumentCloud y comunidades regionales de la mayor red de periodistas y desarrolladores del mundo.

Participamos en conjunto con otro proyecto final en el que van a utilizar nuestra API para hacer detección de entidades en documentos PDF.

La experiencia fue muy satisfactoria, recibimos buenas críticas sobre la Usabilidad de nuestra aplicación y la gran utilidad que presta a la comunidad.

Por tal motivo recibimos el primer premio de dicha hackaton ("Mención itba," 2019)

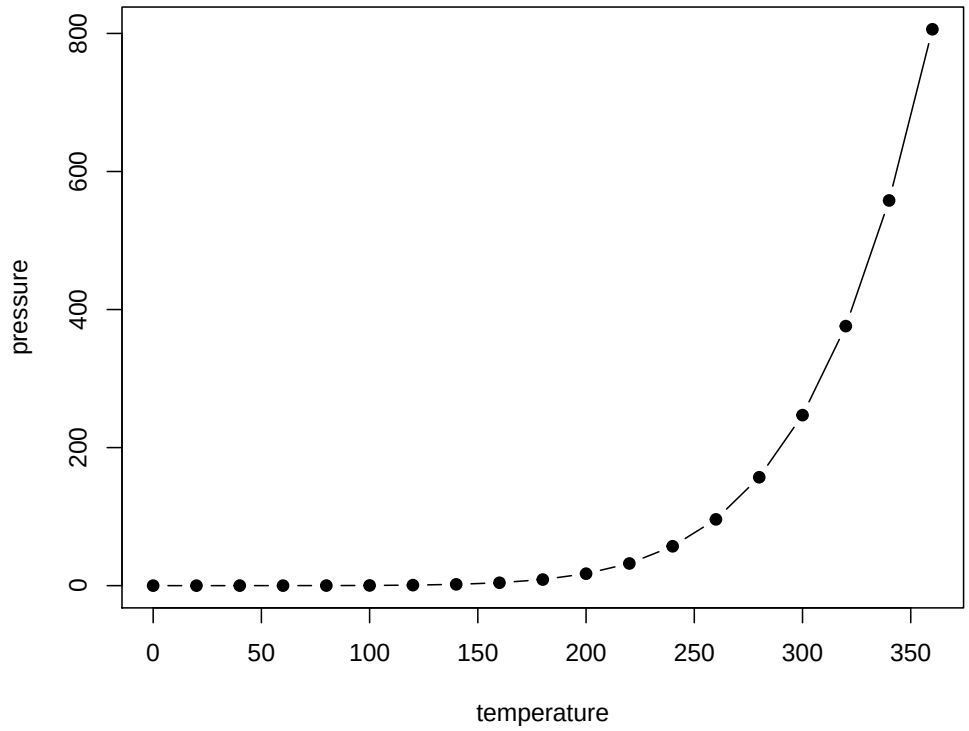


Figura 26: Here is a nice figure!

7. Conclusiones

7.1. Examples

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 1. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter 3.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 26. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2019) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Cuadro 1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

Brunstein, A. (2002). Annotation guidelines for answer types. Retrieved from https://catalog.ldc.upenn.edu/docs/LDC2005T33/BBN-Types-Subtypes.html

Duh definition. (2019). Retrieved October 14, 2019, from https://dictionary.cambridge.org/es/diccionario/ingles/duh

Ethayarajh, K., Duvenaud, D., & Hirst, G. (2019). Towards understanding linear word analogies. *Proceedings of the 57th annual meeting of the association for computational linguistics*, 3253–3262. https://doi.org/10.18653/v1/P19-1315

Hackaton. (2019). Retrieved August 31, 2019, from https://mediaparty.info/

Honnibal, M. (2017). Practical and effective neural ner. Retrieved November 2, 2017, from https://github.com/explosion/talks/blob/master/2017-11-02_Practical-and-Effective-Neural-NER.pdf

Kripke, S. (1980). *Naming and necessity*. Retrieved from https://books.google.com.ar/books?id=9vvAlOBfq0kC

Kruchten, P. (1995). The 4+1 view model of architecture. *IEEE Softw.*, *12*(6), 42–50. https://doi.org/10.1109/52.469759

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. *CoRR*, *abs/1603.01360*. Retrieved from http://arxiv.org/abs/1603.01360

Marsh, E., & Perzanowski, D. (1998). MUC-7 evaluation of IE technology: Overview of results. *Seventh message understanding conference (MUC-7): Proceedings of a conference held in fairfax, virginia, April 29 - may 1, 1998*. Retrieved from https://www.aclweb.org/anthology/M98-1002

Mención itba. (2019). Retrieved October 3, 2019, from https://www.instagram.com/p/B3Koum2peD-/

Montani, I. (2016). Practical and effective neural ner. Retrieved November 28, 2016, from https://github.com/explosion/talks/blob/master/2016-11-28_The-State-of-AI-2016.pdf

Poibeau, T., & Kosseim, L. (2000). Proper name extraction from non-journalistic texts. *CLIN*.

Precision and recall. (2019). Retrieved October 16, 2019, from https://en.wikipedia.org/wiki/Precision_and_recall

The state of the octoverse: Machine learning. (2019). Retrieved January 24, 2019, from https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Retrieved from http://yihui.name/knitr/

Xie, Y. (2019). *Bookdown: Authoring books and technical documents with r markdown*. Retrieved from https://github.com/rstudio/bookdown