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

2019-10-16

Índice

Abstract

TODO: escribir abstract

1. Introducción

2. Estado del arte

2.1. El «naive approach»

tagueo con expresiones regulares...

2.2. Redes neuronales y modelos estadísticos

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

2.3. La formula para «deep-learning»

articulo spacy

2.3.1. embed

Figura 1: TODO: embed

2.3.2. encode

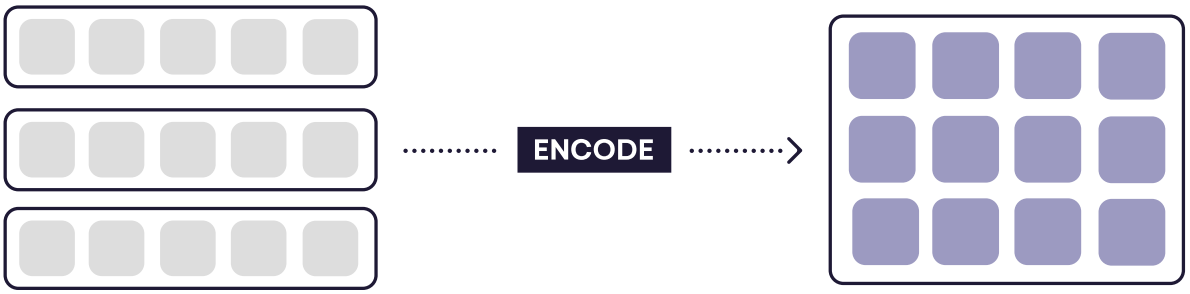

Figura 2: TODO: encode

2.3.3. attend

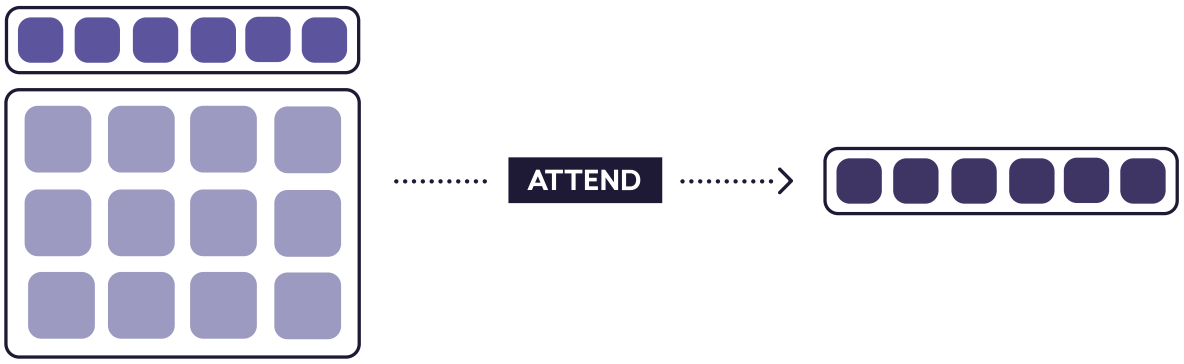

Figura 3: TODO: attend

2.3.4. predict

Figura 4: TODO: predict

Here is a review of existing methods.

2.4. statistical entity recognition model

2.5. Word vectors

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

(Ethayarajh, Duvenaud, & Hirst, 2019)

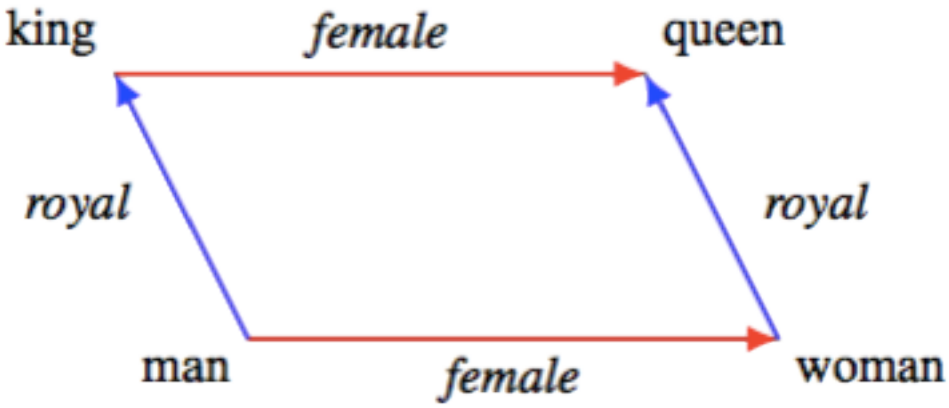

Figura 5: Parallelogram structure in the vector space (by definition)

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/2017-11-02_Practical-and-Effective-Neural-NER.pdf https://github.com/explosion/talks/blob/master/2018-04-12_Embed-Encode-Attend-Predict.pdf

3. Definición del problema

El bottleneck en AI es la data, no los algoritmos. quote de: https://github.com/explosion/talks/blob/master/2016-11-28_The-State-of-AI-2016.pdf

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. 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.

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

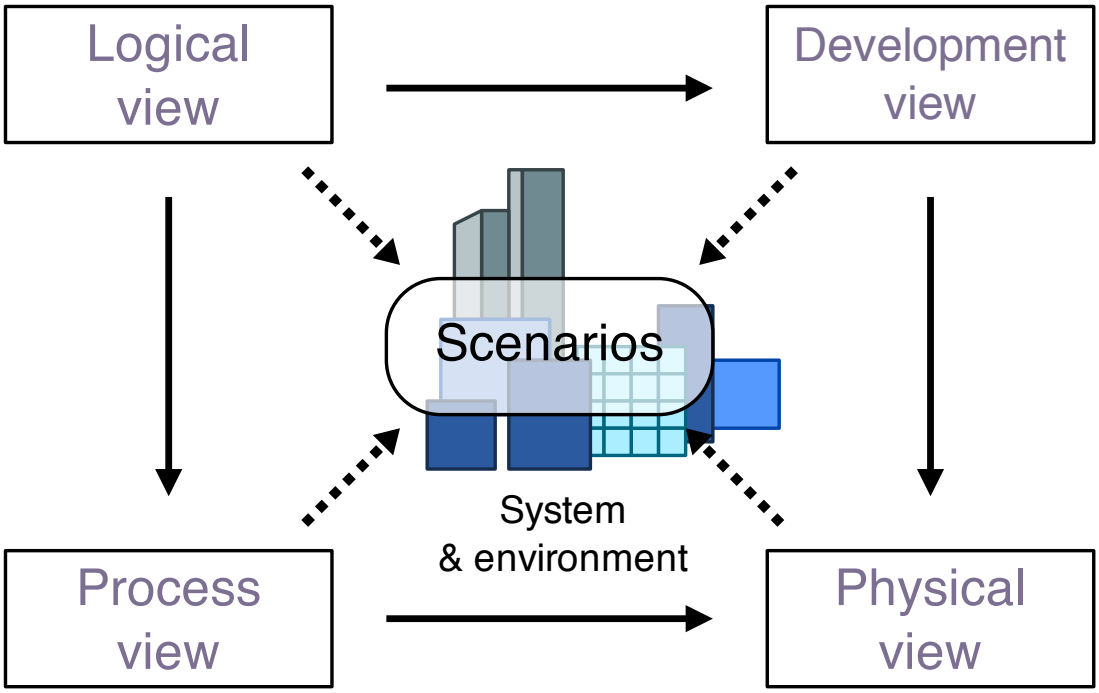

Figura 6: Ilustración de arquitectura 4+1

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.

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.

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.

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). Su implementación es robusta y orientada a la implementción de apliciones en producción, a diferencia de muchas otras librerías de NLP que sólo se utilizan con fines académicos.

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

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. Mejora live vs offline

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

6.4. 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)

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 2.

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 7. 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
)
```

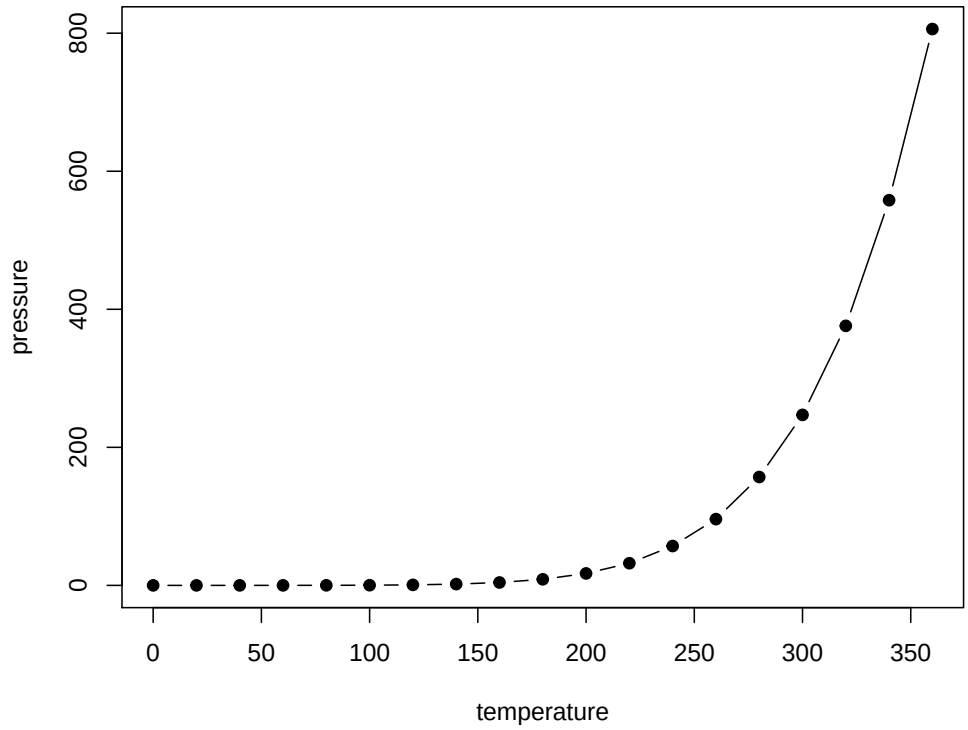

Figura 7: Here is a nice figure!

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

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).

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/

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

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

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