Preliminares para la anotación: El aprendizaje automático

Fernando Carranza
fernandocarranza86@gmail.com

Clase 1
Sábado 22/03/2025

- **Unidad 1: Introducción**
  - **i) la inteligencia artificial,**
  - **ii) programación clásica vs. el aprendizaje automático (machine learning).**
  - **iii) Procesamiento del lenguaje natural: definición, tareas y enfoques;**
  - **iv) lenguaje de programación Python en Google Colab,**
  - v) Exploración de algunos recursos de libre acceso relevantes: Kaggle y Huggingface, GitHub.
- Unidad 2: Los algoritmos supervisados
- Unidad 3: Anotación morfológica y de clase de palabra
- Unidad 4: Anotación sintáctica
- Unidad 5: Anotación para propósitos específicos

Presentación

Estructura y temas de la clase de hoy:

1. Introducción
2. 1.i) la inteligencia artificial
3. 1.ii) programación clásica vs. el aprendizaje automático (*machine learning*)
4. 1.iii) Procesamiento del lenguaje natural: definición, tareas y enfoques
5. 1.iv) lenguaje de programación Python en Google Colab
6. Recapitulación
7. Bibliografía

Qué es la inteligencia artificial

> *the effort to automate intellectual tasks normally performed by humans.*
>
> *(Chollet, 2018, p. 4)*

La inteligencia artificial nace como disciplina hacia los años 50. Un texto pionero es el de Turing (1950). Vale la pena repasar un poco ese texto.

Turing 1950

- Pregunta: ¿Las máquinas pueden pensar?
- Esto requiere definir qué es una máquina y qué es pensar.
- Dado que esto no es tarea fácil, Turing reformula esta pregunta por lo que él llama el juego de imitación, posteriormente conocido como test de Turing.

Turing 1950

Juego de imitación: una persona y una computadora interactúan con un evaluador. Este evaluador debe decidir, solo a partir de las respuestas que recibe, quién es la persona y quién es la computadora. Si la computadora logra engañar al evaluador, podrá decirse que su comportamiento es en alguna medida indistinguible del de la persona.

Turing 1950

¿Qué entendemos por computadora?
Si bien no lo cita, Turing básicamente define su arquitectura en términos de la arquitectura de von Neumann. En los términos de Turing, esto consta de lo siguiente:

1. Store: Lugar en el que se guarda la información. La memory unit del modelo de von Neumann. Equivale a la RAM en una computadora moderna.
2. Executive unit: La parte encargada de hacer las operaciones. La arithmetic/logic unit de von Neumann.
3. Control unit: La parte encargada de controlar que las operaciones se hagan en el orden y forma correcta.

Para completar la arquitectura faltarían los dispositivos de salida y de entrada.

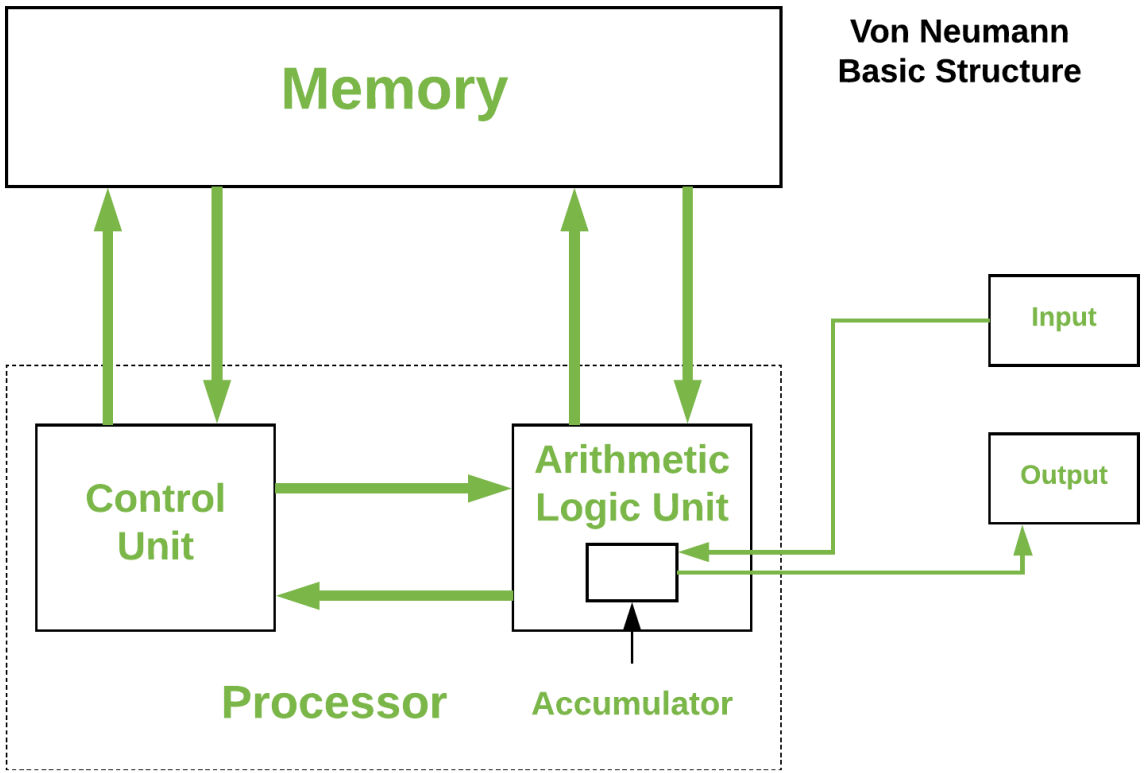

Figura: Arquitectura de von Neumann. Tomada de `https://www.geeksforgeeks.org/computer-organization-von-neumann-architecture/`

Turing (1950)

Turing observa que la construcción de las instrucciones que la unidad ejecutiva debe leer en la memoria para poder operar se conoce como "programar" y define, por lo tanto, que programar una computadora para que haga la operación A consiste en darle las instrucciones para que haga A.

Turing observa que este tipo de computadoras digitales son *máquinas universales*. Esto significa que pueden imitar a cualquier máquina de estados discretos.
Si bien no desarrolla este punto Turing en el artículo, sabemos que las computadoras que responden a la arquitectura de von Neumann y las máquinas de Turing son mutuamente traducibles y, por lo tanto, equivalentes.

- Una Máquina de Turing es una cinta infinita con símbolos, un escaner que lee esos símbolos y un conjunto de estados con instrucciones que definen qué debe hacer en cada momento al ver un símbolo. Estas instrucciones incluyen la posibilidad de reemplazar el símbolo x que se lee por y (sean x e y iguales o no), moverse hacia adelante o hacia atrás en la cinta y pasar del estado q_i al estado q_j (sean q_i y q_j iguales o no)
- Toda función que pueda ser resuelta mediante una MT en una cantidad finita de pasos es una función computable.

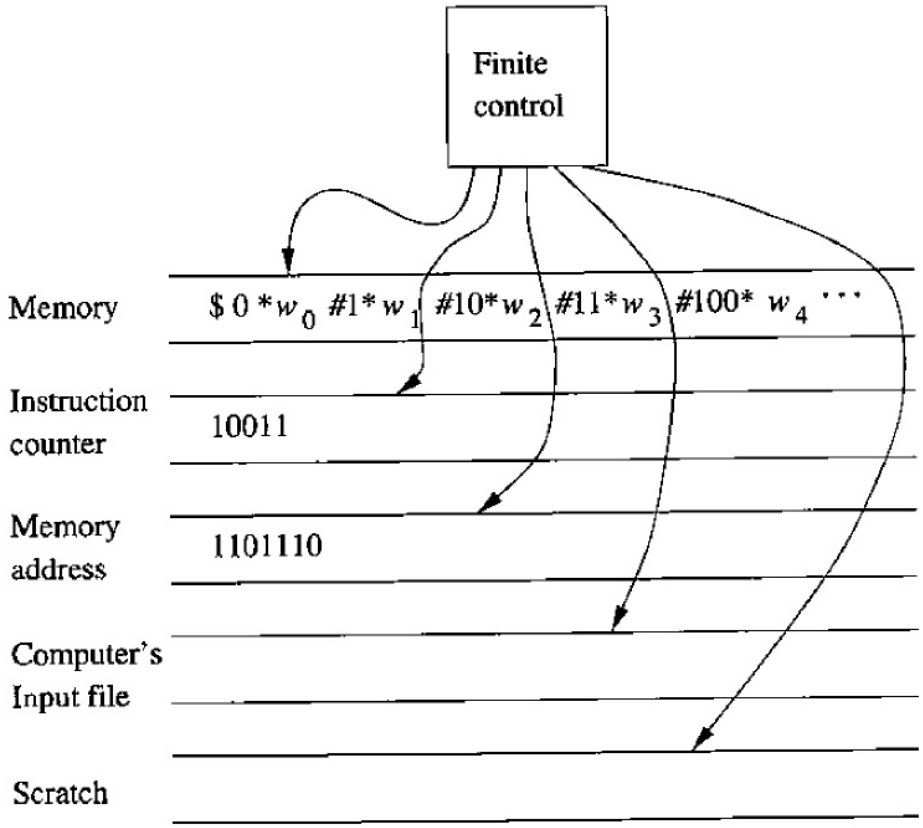

Figura: Tomado de (Hopcroft *et al.*, 2006, p. 359)

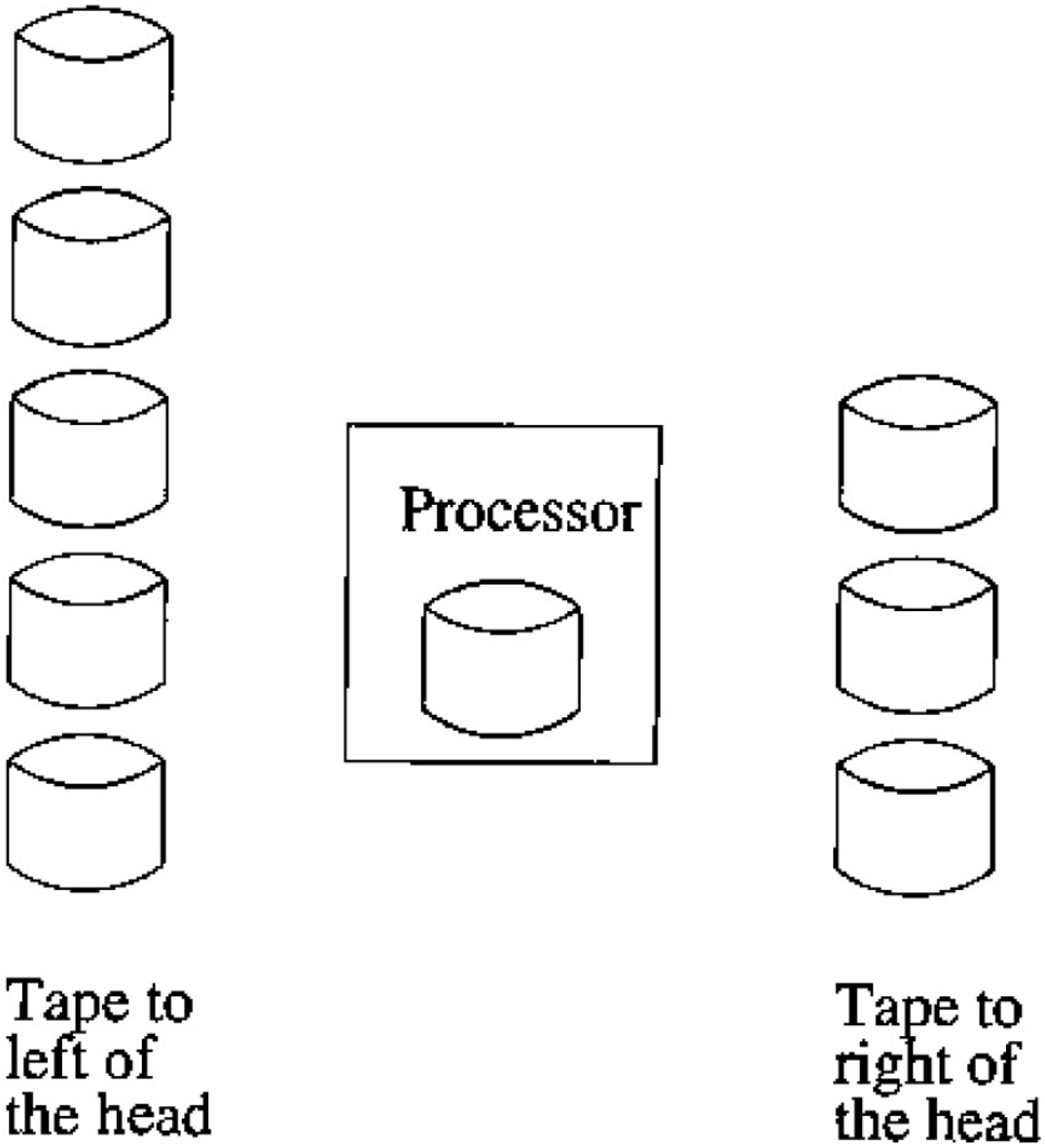

Figura: Tomado de (Hopcroft *et al.*, 2006, p. 356)

- Las computadoras, por ende, son capaces de computar precisamente todo aquello que sea computable.

- Una vez definido lo que es una computadora, Turing redefine su pregunta inicial en términos de si existe una computadora imaginable que pueda ser indistinguible de un ser humano.
- Él responde afirmativamente a la pregunta y en el resto del artículo intenta sostener su hipótesis respondiendo a las objeciones de los escépticos antes que demostrando la verdad de su posición.
- No nos compete entrar en estas objeciones y en las respuestas de Turing, con excepción de una: la objeción de Lady Lovelace.

Turing (1950)

- Objeción de Lady Lovelace: las computadoras solo son capaces de hacer aquello que le indiquemos cómo hacer.
- Turing observa que, en realidad, como ya había observado Hartree, nada impide que eventualmente se construyan máquinas que aprendan. Turing agrega que, cualquier máquina universal tiene la capacidad de aprender.

Turing (1950)

Sin entrar en mayores detalles, nos importan del texto por el momento las siguientes cuestiones:

- El test de Turing.
- Las computadoras digitales como computadoras universales.
- Programar como darle a la computadora una serie de instrucciones para que resuelva tareas
- La posibilidad de que la computadora aprenda a realizar las instrucciones que debe resolver.

Chollet (2017)

Chollet (2018) grafica lo que llama el modo de programación clásico, es decir, dotar a la computadora de instrucciones para que resuelva la tarea que se busca que resuelva, de la siguiente forma:

Figura: Programación clásica

La programación clásica da una respuesta confiable en la medida en que quien haya hecho la programación haya previsto en el código exactamente todo lo que pudiera encontrarse el algoritmo.

- Por ejemplo, un lematizador es un algoritmo que toma una palabra y devuelve su lema. Suponiendo que en el español no existieran palabras homónimas, bastaría una lista L que empareje palabras con lemas y un algoritmo A que sea capaz de buscar por cada palabra de un dato de entrada D, su equivalente en la columna de palabras de la lista L y devuelva su respectivo lema para poder resolver correctamente la tarea.
  Algoritmos similares podrían hacerse para buscar las raíces (*stems*) o las etiquetas de clases de palabras (*pos tags*), si no tuviéramos el problema de la homonimia o el sincretismo.

Este tipo de algoritmos se pueden complementar con uno que decida qué lema, raíz o etiqueta de palabra usar en caso de ambigüedad. Puede encontrarse una lista para hacer este algoritmo en `https://github.com/TALP-UPC/FreeLing/tree/master/data/es/dictionary/entries`

- Supongamos ahora que quisiéramos traducir de una lengua a otra. Esta tarea se conoce con el nombre de *machine translation*.
- En el pasado, se ha intentado resolver esta tarea mediante la declaración de equivalencias, por ejemplo, a través de gramáticas transductoras (ver Roark y Sproat 2007: 279-283).

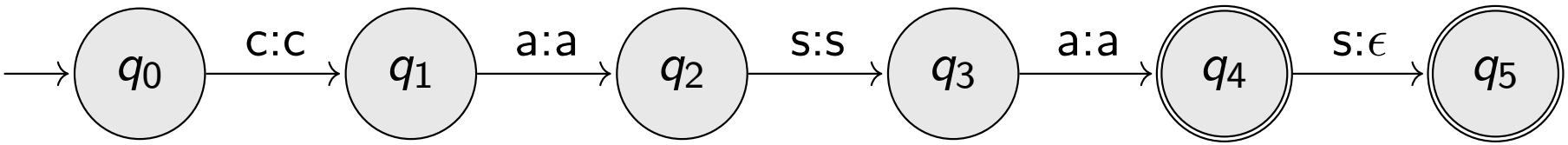

Figura: Transducer que devuelve el lema de casa o casas

Supongamos esta gramática de juguete:

1. S -> NP VP {NP VP}
2. NP -> D N {D N}
3. N -> doctor {doctora}
4. N -> pacient {paciente}
5. NP[a] -> NP {a NP}
6. D -> the {la}
7. VP -> V NP[a] {V NP[a]}
8. V -> saw {vio}

¿Qué equivalencias entre oraciones es capaz de reconocer/producir esta gramática transductora?

Resulta obvio que escribir un algoritmo realista de este tipo que sea capaz de traducir de una lengua a otra es una tarea bastante complicada.
De hecho, en 1966 un grupo de científicos financiados por el gobierno de Estados Unidos para desarrollar un algoritmo de *machine translation* para traducir del inglés al ruso y viceversa tuvo que escribir un informe sobre el estado de la cuestión y la posibilidad o no de éxito de esa tarea. El resultado fue el informe ALPAC (`https://www.mt-archive.net/50/ALPAC-1966.pdf`), cuyo escepticismo hizo que el gobierno de Estados Unidos retirara su financiamiento para el área.

Hoy en día, se avanzó mucho en la traducción automática utilizando para eso modelos estadísticos. Sin embargo, sigue habiendo algunos problemas:

1. Prueben entrar a la siguiente página: `https://lmarena.ai/`
2. Den al chat bot la siguiente instrucción:

   Please translate the following sentence into Spanish: The doctor asked the nurse to prepare the patient for the operation.

- Ahora bien, como ya prefiguraba Turing, nada impide que un algoritmo aprenda. Este tipo de programación es lo que hoy en día se conoce como aprendizaje automático (*Machine Learning*).
- En el aprendizaje automático, el sistema es entrenado antes que programado.

Chollet (2018) grafica este método de programación de la siguiente manera:

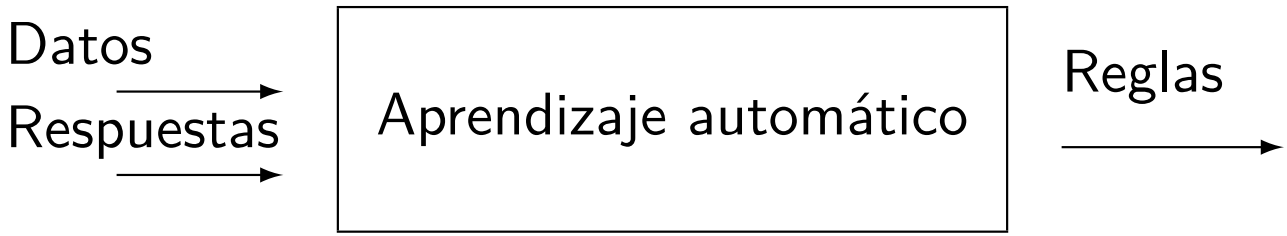

Figura: Aprendizaje automático

Este paradigma es tan antiguo como el paradigma de la programación clásica. El concepto de "aprendizaje automático" fue acuñado por Arthur Samuel en 1959.

> *Samuel describe el aprendizaje automático como una tecnología que utiliza técnicas estadísticas y algoritmos computacionales para proporcionar a los ordenadores la capacidad de aprender, es decir, mejorar sus resultados en una tarea específica tras procesar datos en suficiente cantidad y sin instrucciones explícitas externas proporcionadas por el programador.*
>
> *(Álvarez Vega et al. 2020)*

Sin embargo recién comenzó a ser el paradigma dominante a partir de los años noventa.
Razones según (Moreno Sandoval, 1998, p. 155):

1. Los ordenadores son considerablemente más potentes (...)
2. La accesibilidad a los datos en forma de corpus electrónicos en la actualidad [años 90] es muy superior a décadas anteriores (...)
3. Hay presión por parte de las instituciones que costean las investigaciones en LC [Lingüística computacional] para obtener resultados utilizables (...). La "robustez" (la capacidad para no quedarse atascado cuando el sistema se enfrenta a un problema, es decir, una construcción o una palabra desconocida) de los métodos estadísticos es mayor.

Para el aprendizaje automático se necesitan tres cosas:

- Datos de entrada (*input data points*)
- Ejemplos del resultado esperado (*Examples of the expected output*)
- Un modo de medir la tasa de acierto (*A way to measure whether the algorithm is doing a good job*)

La lingüística computacional (LC) es una ciencia interdisciplinaria que se ubica entre la linguística y la informática, con énfasis en la lingüuística. Su fin es la elaboración de modelos computacionales que reproduzcan uno o más aspectos del lenguaje humano. Dos áreas aledañas a la LC son el procesamiento del habla realizado por parte de la informática y el reconocimiento de voz desarrollado por la ingeniería eléctrica.

Domínguez Burgos (2002)

Una historización según Domínguez Burgos (2002):

- años 40-50: Desarrollo de la teoría de los autómatas y de la teoría de la información.
- años 60: informe ALPAC, corpus de Brown, ELIZA.
- años 70: surge PROLOG y comienzan las primeras gramáticas de unificación.
- años 80: toman fuerza los modelos probabilísticos.
- años 90: aparece Internet.

Algunos hitos no contemplados en Domínguez Burgos (2002)

- años 2010: aparecen las redes neuronales
- años 2020: aparecen los grandes modelos de lenguaje

Algunas áreas de la LC

- *Machine translation* o traducción automática
- diseño de *chatbots*, sistemas de diálogo, sistemas expertos
- Etiquetamiento morfológico o *pos-tagging*
- Análisis sintáctico o *parsing*
- *Information retrieval* o recuperación inteligente de información

¿Qué se necesita para dedicarse a la lingüística computacional?

- Conocimientos de programación. Domínguez Burgos (2002) habla de C, C++, PERL, Java y Visual Basic. Advierte que esto se puede desactualizar con el tiempo. Hoy en día, Python se impuso como uno de los más importantes.
- Conocimientos de lógica y matemáticas: Domínguez Burgos (2002) incluye teoría de conjuntos, funciones, autómatas, teoría de grafos y álgebra lineal, estadística y teoría de probabilidades.
- Estructura de datos.
- Herramientas para trabajar en equipo. Hoy, resulta indispensable el manejo de las herramientas colaborativas de google y de git.

Para el punto 1.iv vamos a usar el archivo intro-python.

En esta clase tratamos de introducir los siguientes temas:

- En qué consiste la inteligencia artificial y algunas nociones generales de cómo funcionan las computadoras.
- En qué consiste la programación clásica y el aprendizaje automático.
- Algunas nociones básicas de programación en Python

Bibliografía I

Álvarez Vega, M., Quirós Mora, L. M., y Cortés Badilla, M. V. (2020). Inteligencia artificial y aprendizaje automático en medicina. *Revista Médica Sinergia*, 5(8).

Chollet, F. (2018). *Deep Learning with Python*. Manning, Shelter Island.

Domínguez Burgos, A. (2002). Lingüística computacional. un esbozo. *Boletín de lingüística*, 18:104–119.

Hopcroft, J. E., Motwani, R., y Ullman, J. D. (2006). *Automata theory, languages, and computation*. Addison-Wesley, Boston, Massachusetts.

Moreno Sandoval, A. (1998). *Lingüística computacional. Introducción a los modelos simbólicos, estadísticos y biológicos*. Síntesis, Madrid.

Roark, B. y Sproat, R. (2007). *Computational Approaches to Morphology and Syntax*. Oxford University Press, Oxford.

Turing, A. (1950). Computing machinery and intelligence. *Mind. A quarterly review of psuchology and philosophy*, LIX(236).