

# IA Conversacional ❤ Python

Nieves Ábalos > [@nieves\\_as](https://twitter.com/nieves_as)

Cofundadora y CPO en Monoceros Labs.

 monoceros  
LABS



## Nieves Ábalos Serrano

Chief Product Officer & Cofundadora [Monoceros Labs](#)  
Twitter: [@nieves\\_as](#)  
LinkedIn: [/in/nievesabalosserrano](#)

- **Universidad de Granada:** Ingeniera Informática, Máster y PhD sin terminar en sistemas de diálogo.
- **Dpto Innovación BEEVA (BBVA Next Technologies):** I+D en conversacional y NLP.
- **Monoceros Labs:** emprendedora, estrategia y gestión de producto, diseño conversacional y desarrollo conversacional (lo que toque).
- **Comunidad:**
  - **Women in Voice Spain (Lead),** 
  - **Alexa Champion,** 
  - **AWS Community Builder.** 

**El presente y futuro  
de la Inteligencia  
Artificial está  
marcado por el  
lenguaje humano.**

Y el lenguaje  
humano es  
complejo: símbolos,  
significado,  
contexto...

.. el lenguaje humano  
es impreciso y  
ambiguo.

# Hablando de ambigüedad

... seguro que habéis escuchado alguno de estos conceptos:

Interfaz  
conversacional

CUI

interfaz  
de voz

agente  
conversacional

spoken  
dialogue  
system

voice app

VUI

bot

chatbot

asistente  
virtual

# IA conversacional

 Tecnología  
para la  
**comunicación**  
con las  
personas

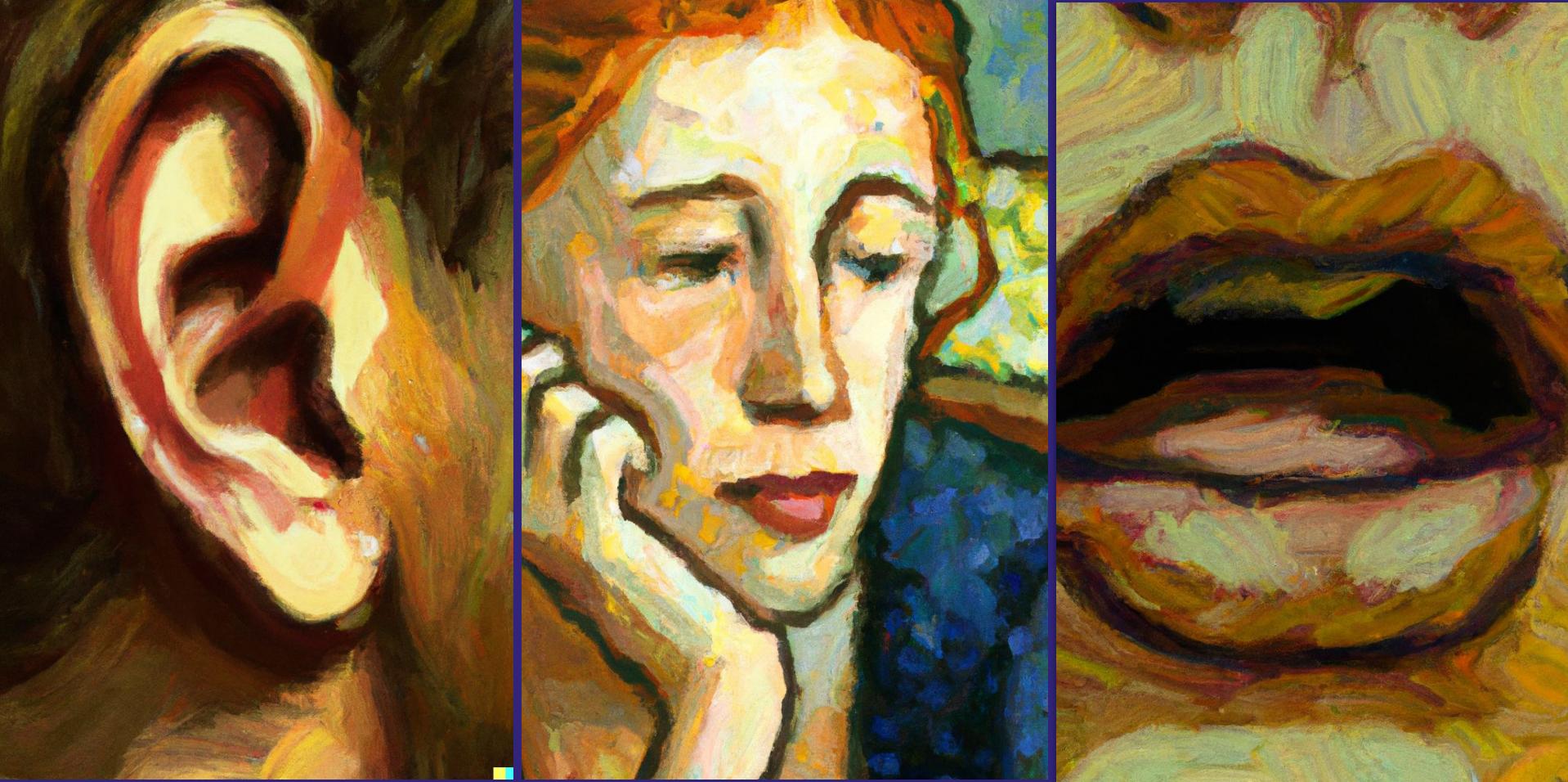
## INTELIGENCIA ARTIFICIAL

CONVERSACIONAL

ML

DL

NLP





# Redes Neuronales

... están cambiando las reglas del juego con respecto a lo que podemos construir.

**Cómo conseguimos  
conversaciones con las máquinas más  
naturales(\*) y efectivas  
... utilizando Python.**

**Junio 2022**

SAN FRANCISCO  
CALIFORNIA  
11:26  LIVE



► BLAKE LEMOINE | GOOGLE AI ENGINEER

**GOOGLE ADVANCEMENTS**  
ENGINEER CLAIMS COMPANY'S AI MACHINES HAVE THOUGHTS

| THE NEWS DESK

19:26 >

**.. ha estado hablando  
con una máquina:  
y ésta es consciente de sí  
misma, tiene sentimientos,  
pensamientos.**



# LaMDA (Mayo 2021)

Language Model for Dialogue Applications

[blog.google/technology/ai/lamda](https://blog.google/technology/ai/lamda)

Es un modelo al que le damos **un objetivo, y buscará conseguirlo a través del diálogo.**

¿LaMDA entiende<sup>(\*)</sup>  
lo que dice?



## LaMDA = modelo del lenguaje:

- **genera** respuestas que nos parece que tienen sentido,
- **aprende posibles respuestas** en base a una **probabilidad**,
- y además esa respuesta la **condicionamos en base al contexto** (lo que le hemos dicho antes).

¿.. un poco loro?

Lemoine habló con un  
modelo del lenguaje  
cuyo objetivo era  
convencerlo de que  
tenía conciencia...

... y la conciencia es  
una cualidad humana.

# Antropo- morfismo

Dar **forma** o  
**cualidades**  
**humanas**  
a lo que no lo es.



¿Qué cualidades  
humanas  
buscamos  
en la tecnología  
conversacional?

# COMUNICAR EN LENGUAJE NATURAL



COMUNICAR  
EN LENGUAJE  
NATURAL



CREAR UN  
VÍNCULO

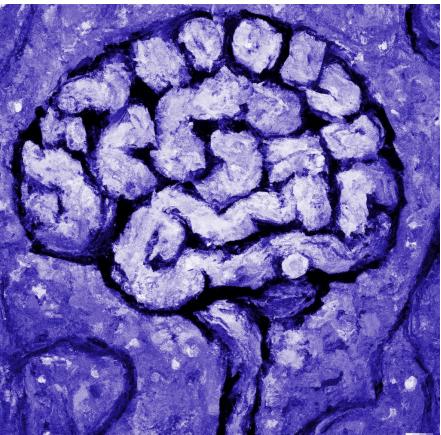
**COMUNICAR  
EN LENGUAJE  
NATURAL**



**CREAR UN  
VÍNCULO**



**INTELIGENCIA**



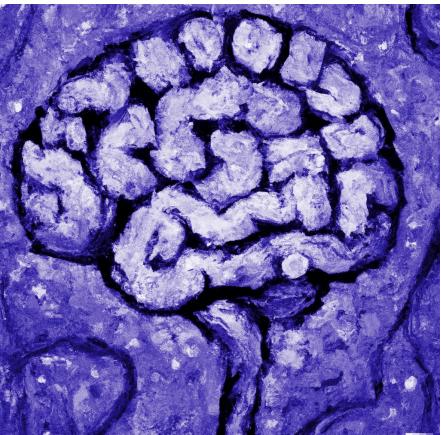
**COMUNICAR  
EN LENGUAJE  
NATURAL**



**CREAR UN  
VÍNCULO**



**INTELIGENCIA**

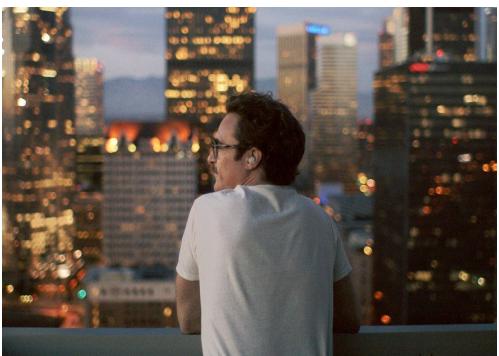
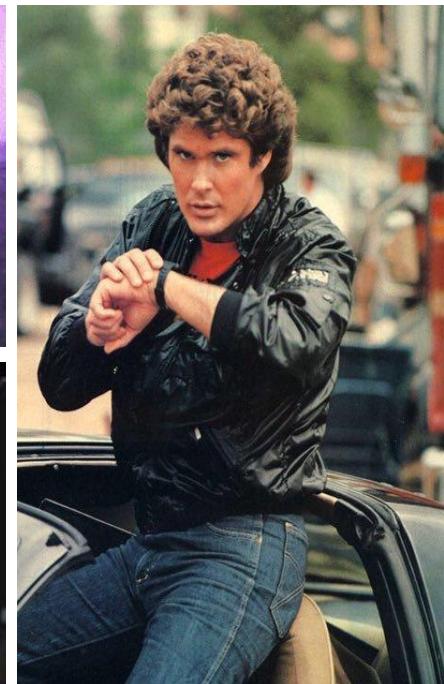
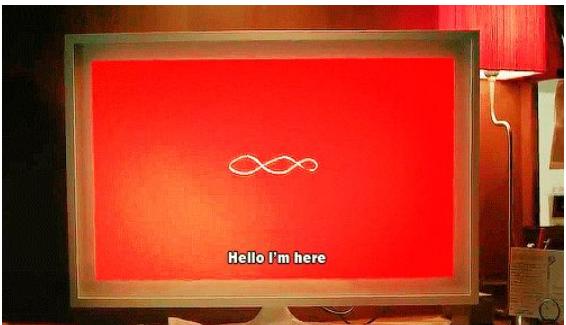


**CONFIANZA**



¿Expectativas  
altas...?

# ¡Ay, la ciencia ficción!

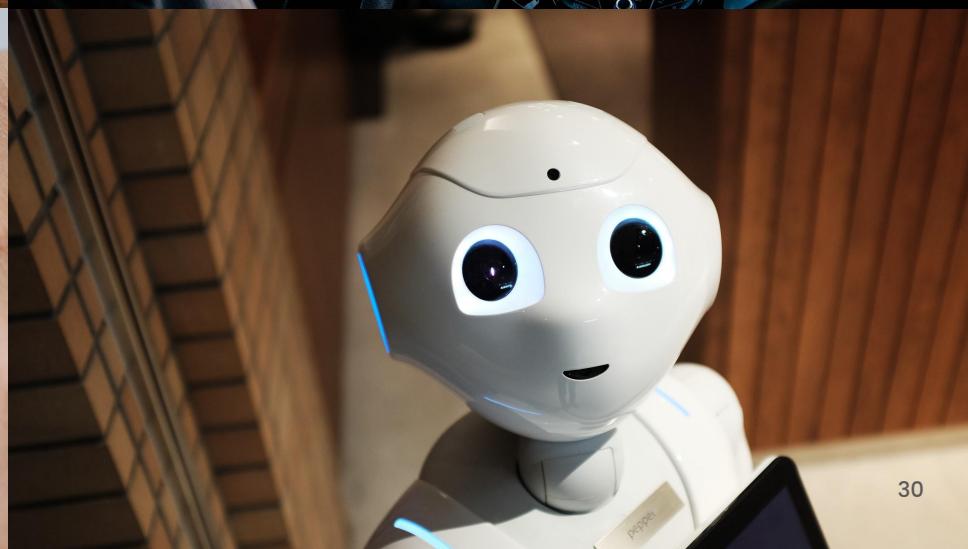


---

## Y luego, la realidad...



Vamos con la realidad:  
qué podemos construir  
con la tecnología que  
tenemos disponible  
(y qué limitaciones tiene)



.. pero no todas las  
conversaciones  
son igual de fáciles  
de construir.

## La complejidad según el tipo de conversación...

Closed-domain  
(un dominio  
concreto)

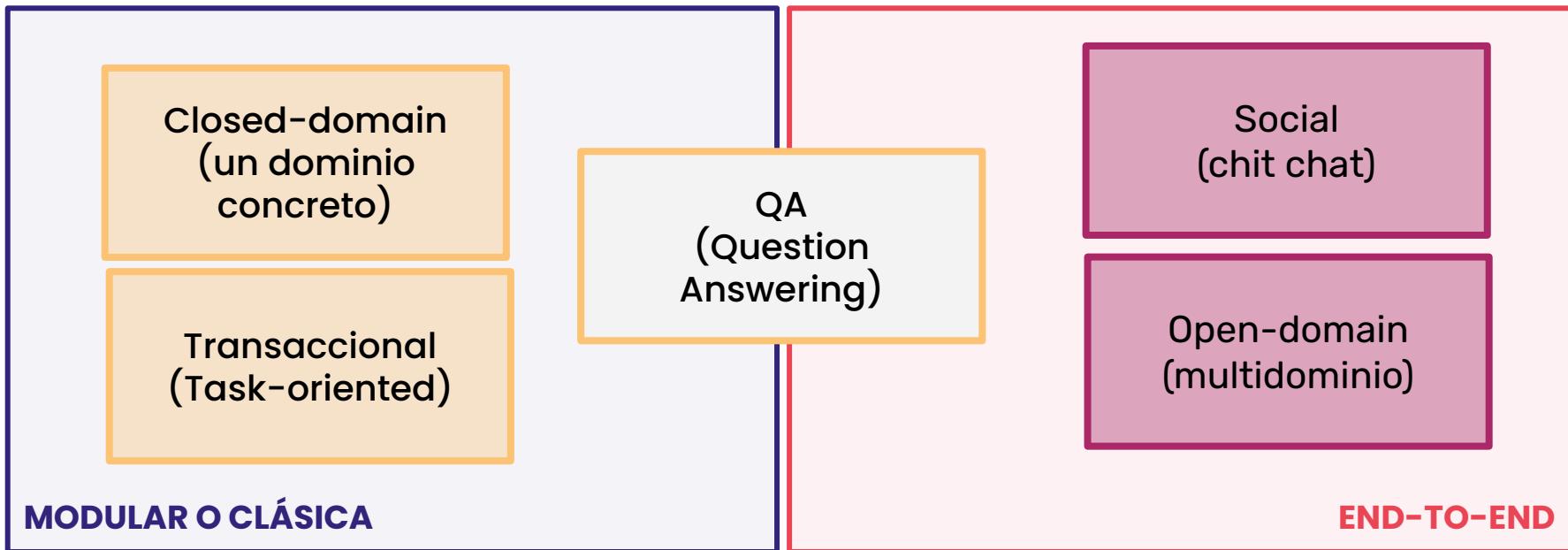
Transaccional  
(Task-oriented)

QA  
(Question  
Answering)

Social  
(chit chat)

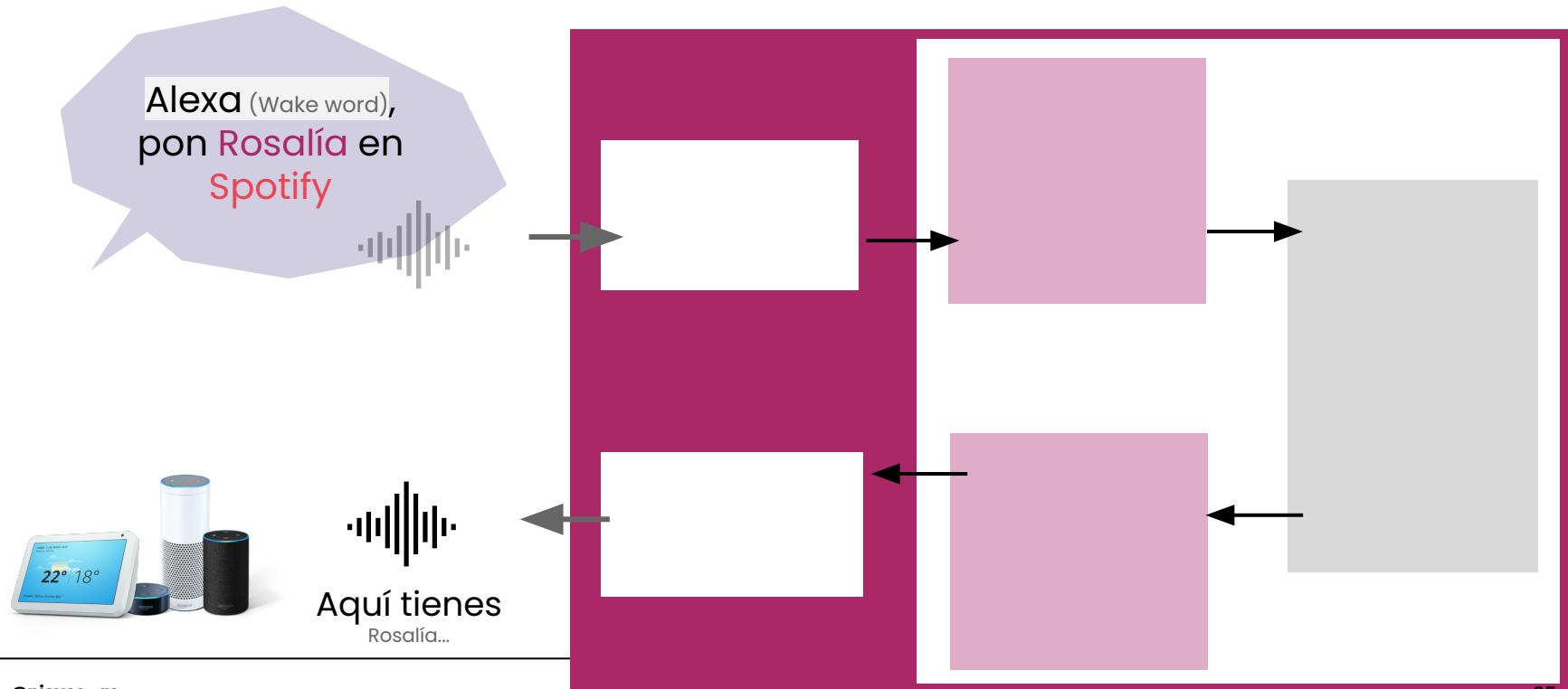
Open-domain  
(multidominio)

.. y (generalizando) según las piezas de la arquitectura.

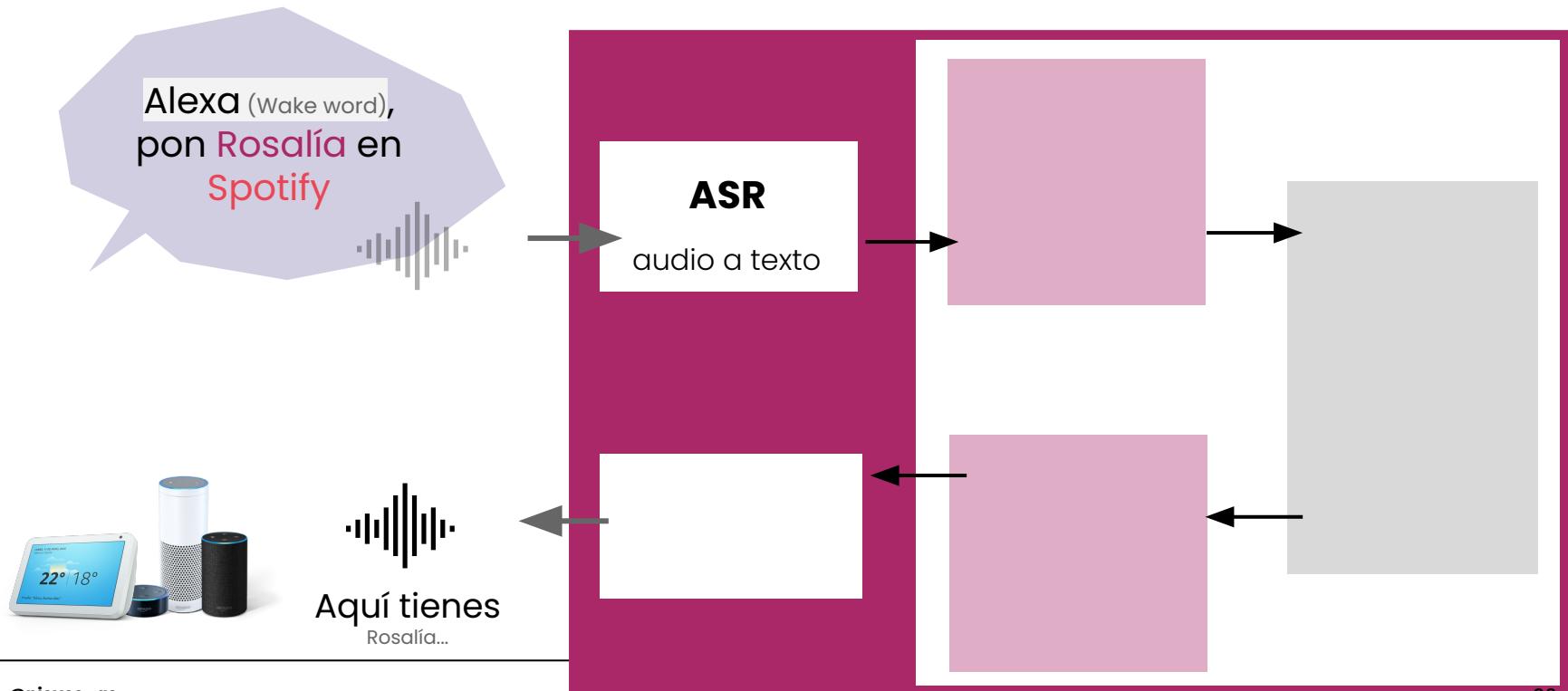


**Arquitectura  
modular o clásica.**

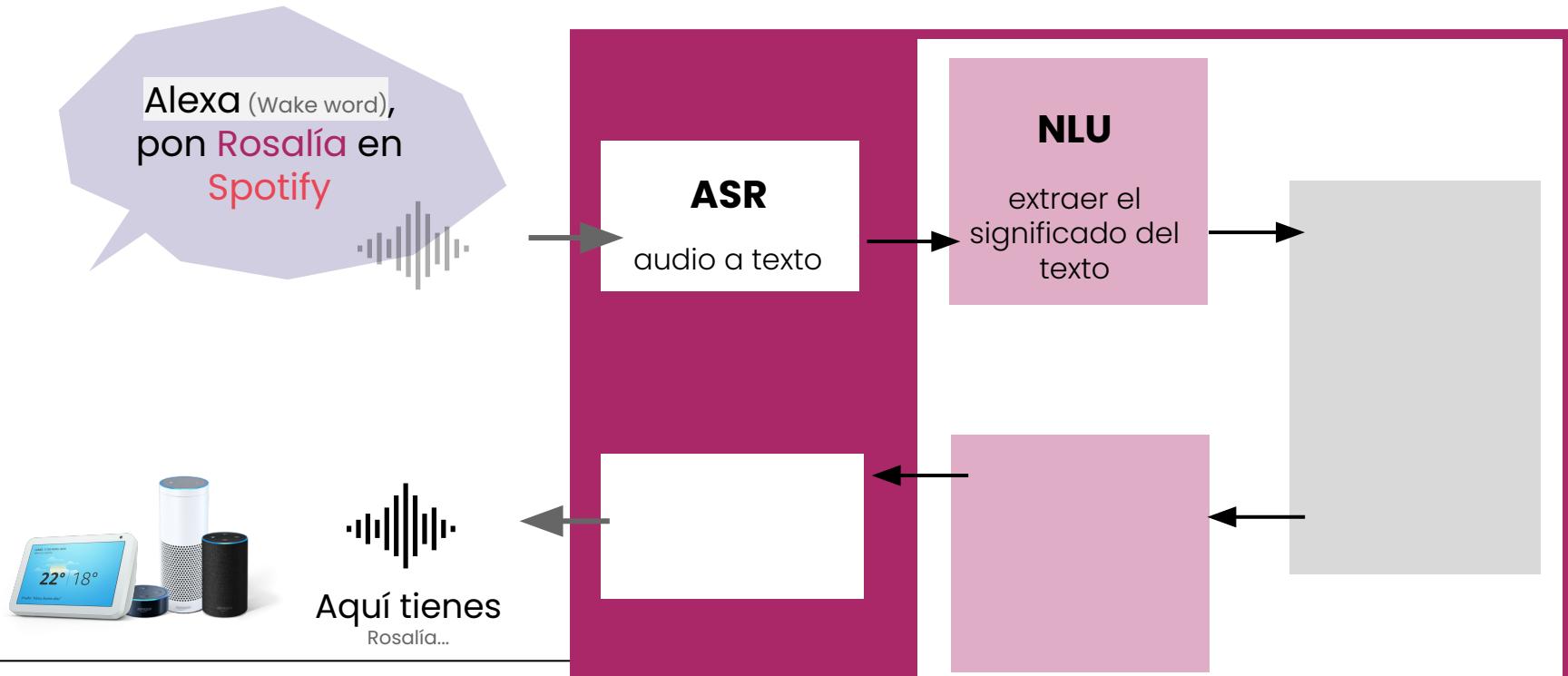
# Los módulos o fases: interfaz de voz



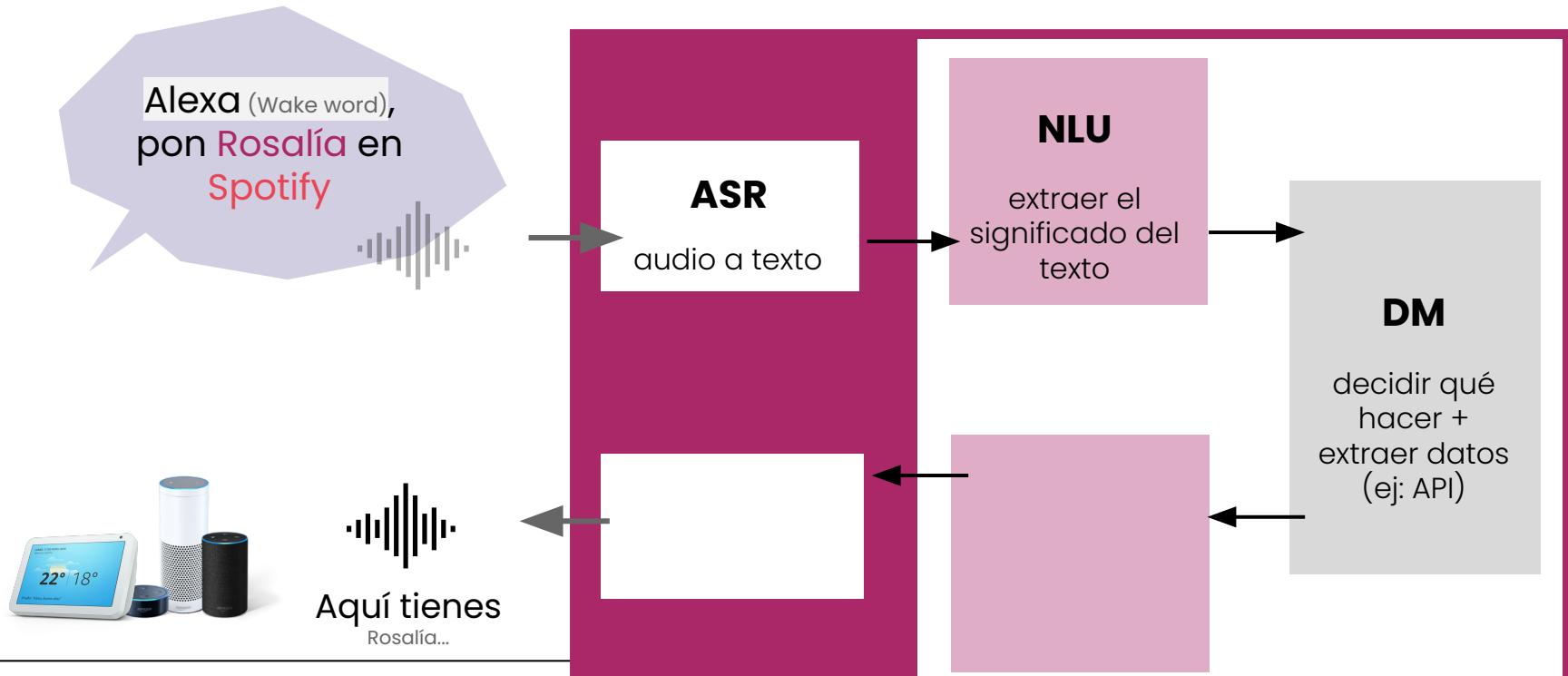
# Los módulos o fases: interfaz de voz



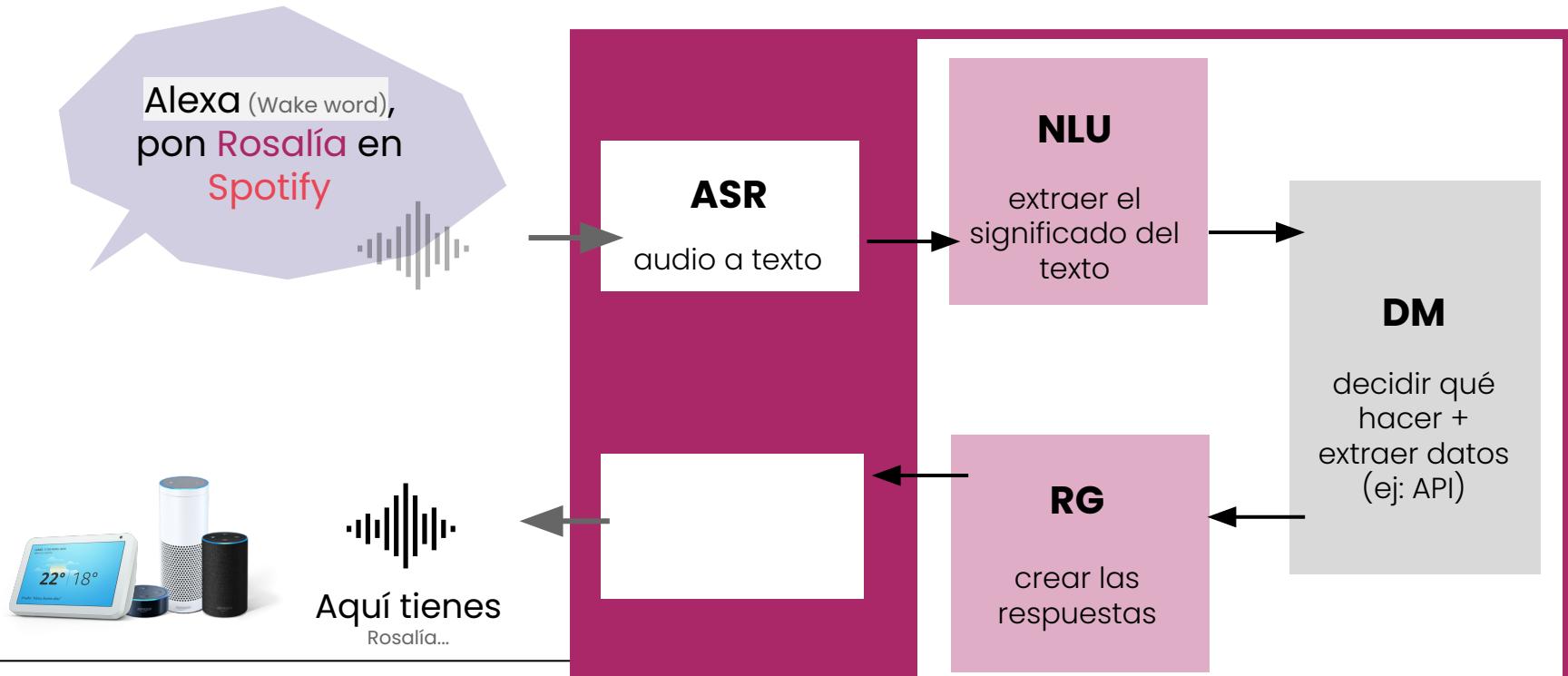
# Los módulos o fases: interfaz de voz



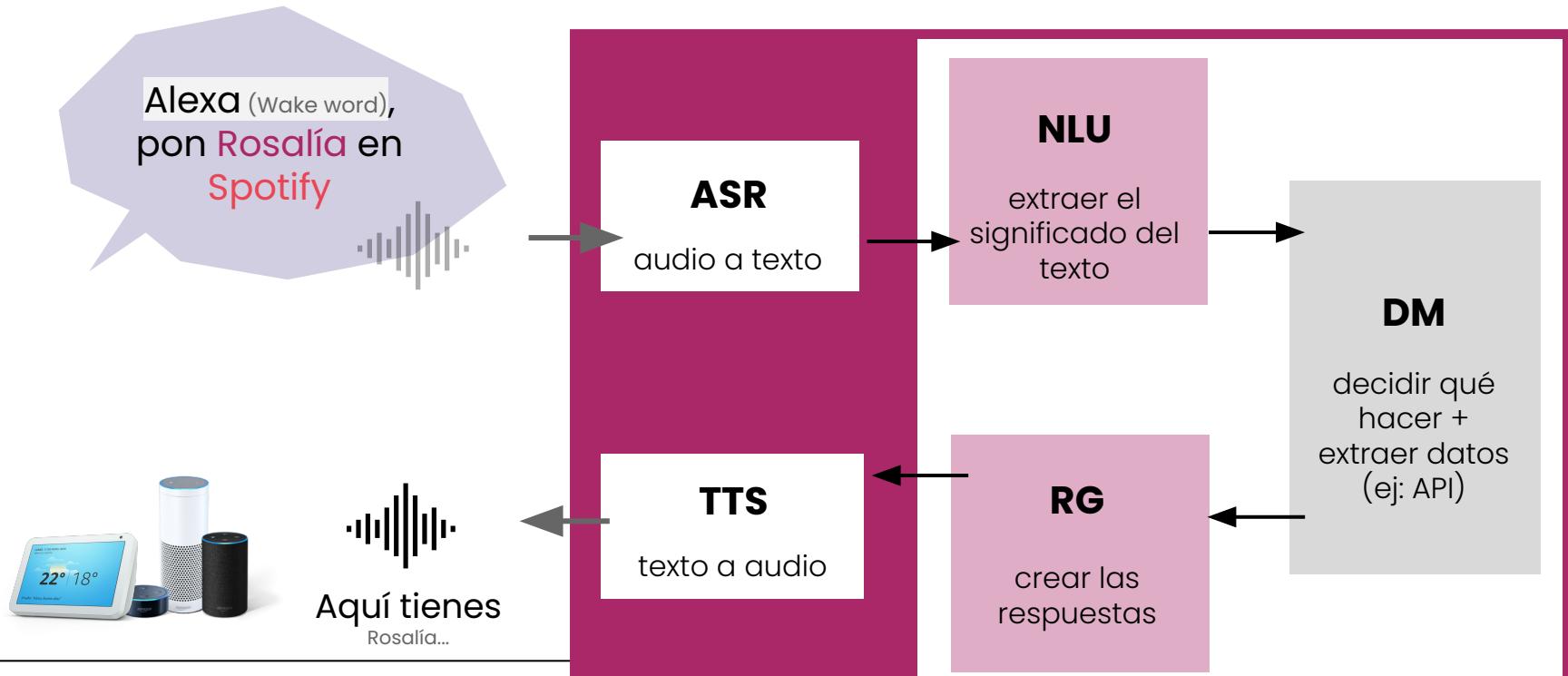
# Los módulos o fases: interfaz de voz



# Los módulos o fases: interfaz de voz

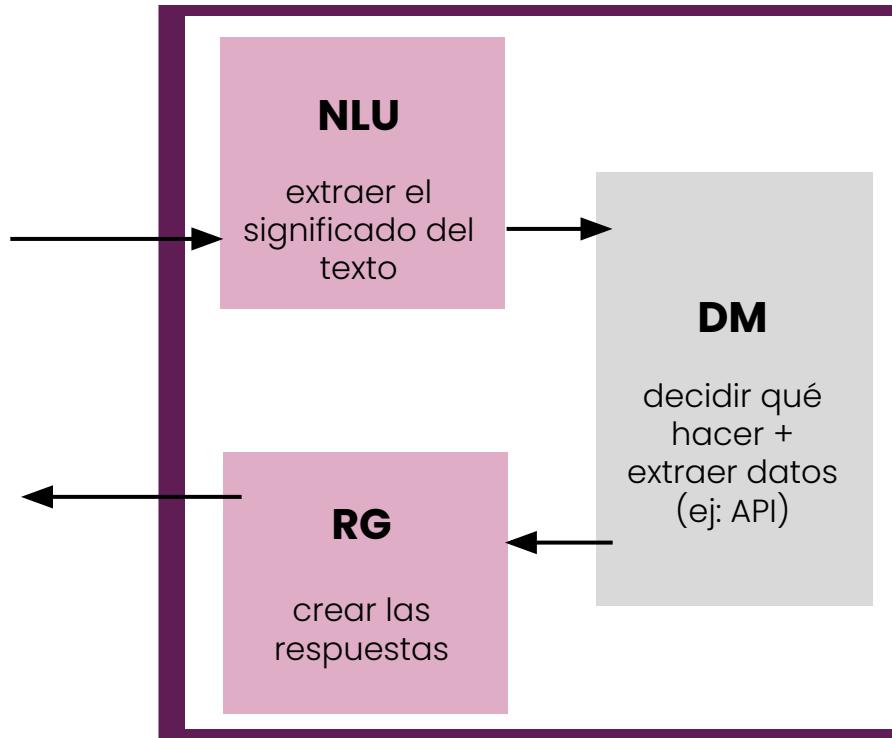
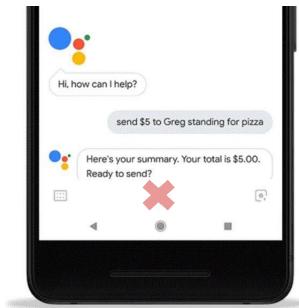


# Los módulos o fases: interfaz de voz



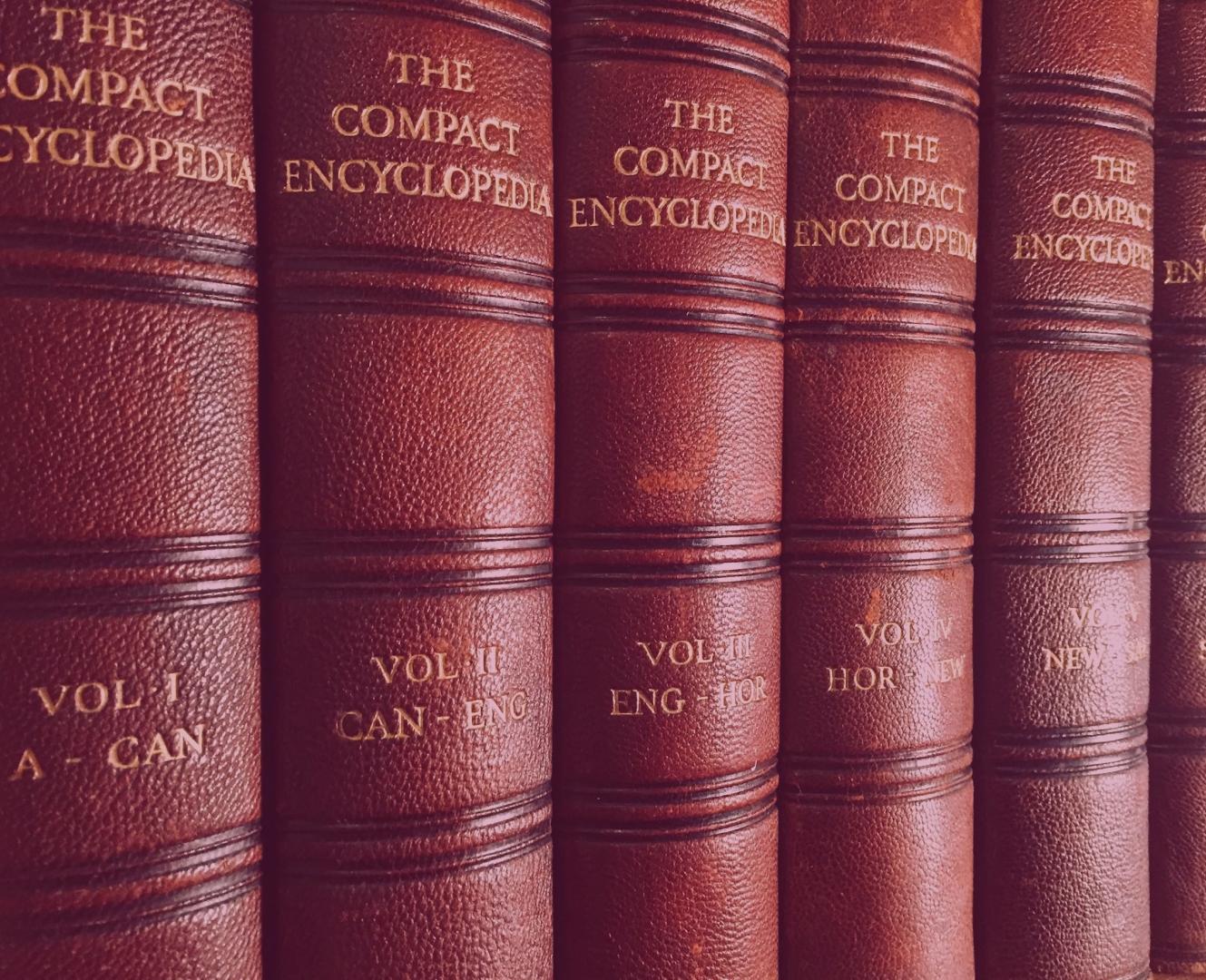
# Los módulos o fases: interfaz de texto

**Escribo:**  
“abre Juego de Tronos”



# NLU

Lo más importante:  
intentar entender el  
significado.  
Esta pieza está  
**presente** en casi  
todas **las**  
**plataformas de voz**  
**y texto.**



# El NLU en estas plataformas es un **modelo** que contiene...



El NLU contiene:

- **todas** las *frases de entrenamiento* del dominio, clasificadas de la manera más óptima en las *intenciones* (o cajas), y
- **todas** las *entidades* con los *valores* que podemos entender.



NLP AVANZADO  
con spaCy

### Capítulo 1: Encontrando palabras, frases, nombres y conceptos

Este capítulo te mostrará las bases del procesamiento de texto con spaCy. Aprenderás acerca de las estructuras de datos, cómo trabajar con pipelines entrenados y cómo usar esas estructuras y pipelines para predecir características lingüísticas en tu texto.

### Capítulo 2: Análisis de datos a gran escala con spaCy

En este capítulo usarás tus nuevas habilidades para extraer información específica de grandes volúmenes de texto. Aprenderás a sacarle el mayor provecho a las estructuras de datos de spaCy y cómo combinar los enfoques estadísticos y basados en reglas de manera efectiva para el análisis de texto.

### Capítulo 3: Pipelines de procesamiento

En este capítulo aprenderás todo lo que necesitas saber sobre el pipeline de procesamiento de spaCy. Aprenderás lo que sucede cuando procesas un texto, cómo escribir tus propios componentes y añadirlos al pipeline y cómo usar atributos personalizados para añadir tus propios metadatos a los documentos, spans y tokens.

Capítulo 4: Entrenando un modelo de red neuronal

## Otras herramientas NLU / NLP



wit.ai

**Duckling**  
Github Repository

[Try it out](#)

Introduction  
Limitations  
Getting Started  
Implementation  
Extending Duckling  
Debugging  
Contributing  
FAQ

The linguist that parses text into structured data

[Try it out with a date/time](#)

e.g. tomorrow at 6am

EN Try me!

### Introduction

Duckling is a Clojure library that parses text into structured data:

```
"the first Tuesday of October" => {:value "2014-10-07T00:00:00.000-07:00"  
:grain :day}
```

See our [blog post announcement](#) for more context.

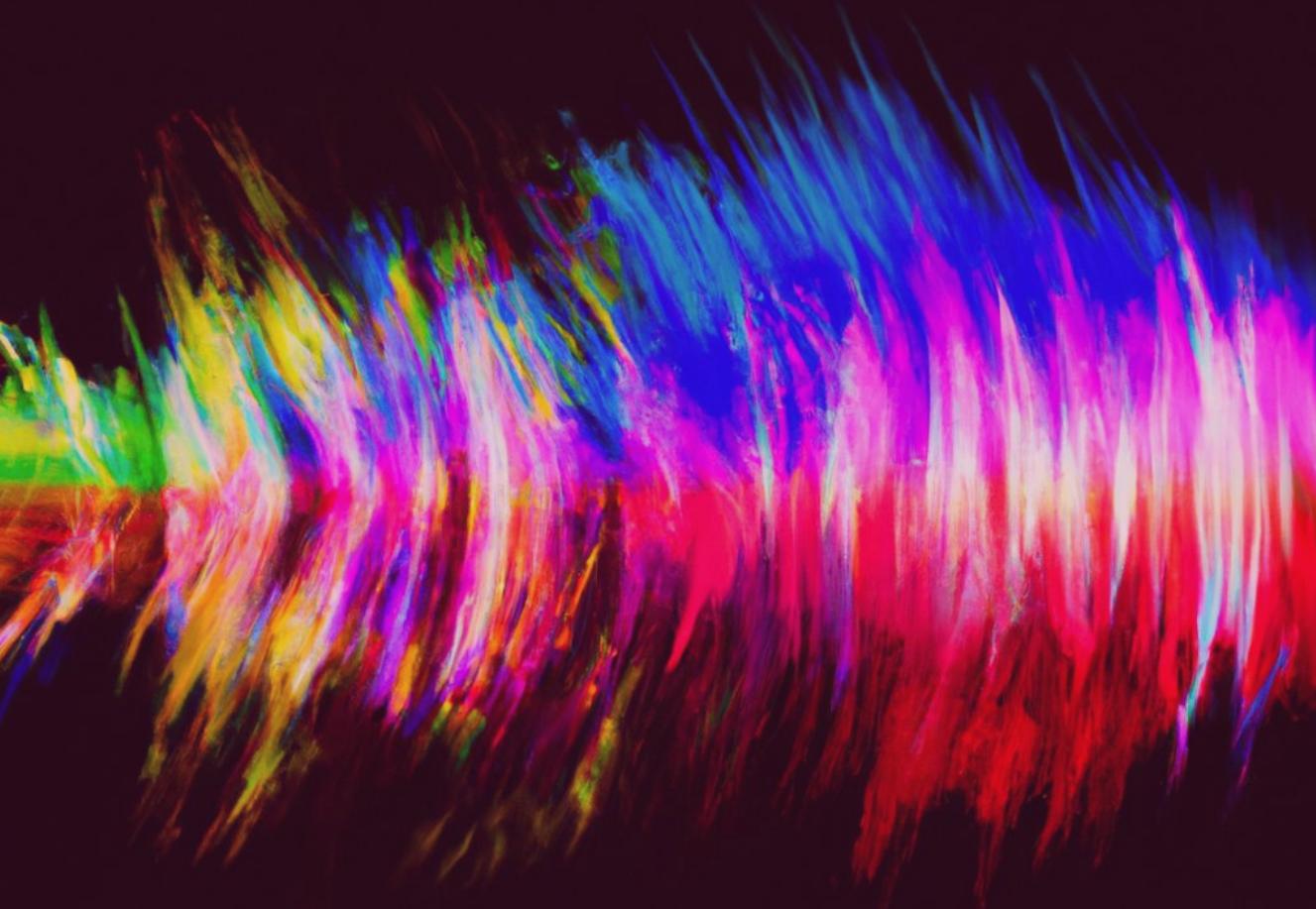
Duckling is shipped with modules that parse temporal expressions in English, Spanish, French, Italian and Chinese (experimental, thanks to Zhe Wang). It recognizes dates and times described in many ways:

- today at 5pm
- 2014-10-01
- the last Tuesday of October 2012
- twenty five minutes ago
- the day before labor day 2020
- June 10-11 (interval)
- third monday after christmas 1980



# TTS

De *Loquendo* a  
voces neurales.



# Voces sintéticas



# KITTS: clonación y voces que no existen.



Voz clonada  
Nieves



Voz que no existe  
[www.estavoznoexiste.com](http://www.estavoznoexiste.com)



# La primera voz sin género



# ASR

Para transcribir de voz a texto.

También llamado *speech to text*.



# El ruido y *The Cocktail Party Problem*

El reto del ASR: tratar de identificar al hablante y separar lo que dice, en entornos de mucho ruido (como una fiesta).



# Whisper (OpenAI, 2022)

open-source

## Introducing Whisper

We've trained and are open-sourcing a neural net called Whisper that approaches human level robustness and accuracy on English speech recognition.

September 21, 2022  
7 minute read

[READ PAPER](#)

[VIEW CODE](#)

[VIEW MODEL CARD](#)

Whisper examples:

Speed talking ▾



[REVEAL TRANSCRIPT](#)

Whisper is an automatic speech recognition (ASR) system trained on 680,000 hours of multilingual and multitask supervised data collected from the web. We show that the use of such a large and diverse dataset leads to improved performance across multiple languages and dialects. Moreover, it is

# Whisper (OpenAI, 2022)

```
audio = "/home/carlos/Música/7db883e9-ab55-47df-9924-c1cb63a8941d.mp3"

result = transcribe_audio(audio)

ipd.display(ipd.Audio(audio, rate=22050, normalize=True))
print(result["text"])
print()
write_vtt(result["segments"])
```

Detected language: spanish



Es hora de crear conversaciones. Somos Monoceros, un estudio de innovación especializado en estrategia, diseño conversacional y tecnologías del habla. Creamos aplicaciones de voz para asistentes como Amazon Alexa y Google Assistant. Además, tenemos un proyecto de investigación tecnológica para crear voces sintéticas más naturales y expresivas en español. El producto se llamará Voces.ai y lo publicaremos este año 2022. Esta voz que escuchas está generada con esta tecnología. Para conseguir este resultado solo hemos necesitado 200 frases de nieve.

Subtítulos WEBVTT

00:00.000 --> 00:02.000

Es hora de crear conversaciones.

00:02.000 --> 00:10.000

Somos Monoceros, un estudio de innovación especializado en estrategia, diseño conversacional y tecnologías del habla.

00:10.000 --> 00:15.000

Creamos aplicaciones de voz para asistentes como Amazon Alexa y Google Assistant.

00:15.000 --> 00:23.000

Además, tenemos un proyecto de investigación tecnológica para crear voces sintéticas más naturales y expresivas en español.

00:23.000 --> 00:29.000

El producto se llamará Voces.ai y lo publicaremos este año 2022.

00:29.000 --> 00:33.000

Esta voz que escuchas está generada con esta tecnología.

00:33.000 --> 01:01.000

Para conseguir este resultado solo hemos necesitado 200 frases de nieve.

# Asistentes y plataformas

Propietarias y  
open-source



## Plataformas para crear apps o asistentes

### Propietarias



amazon alexa



(\*) app actions



Bot Framework SDK



Dialogflow

*Voiceflow*



IBM Watson Assistant



### Open-source



MYCROFT AI



Home Assistant

# Alexa (Hosted-Skill)

Crea una Alexa Skill con Python  
desde la Alexa Developer  
Console.

The screenshot shows the Alexa Developer Console interface. At the top, there are tabs for 'Your Skills' (selected), 'Build', 'Test', 'Distribution', and 'Certification'. Below this, a dropdown menu shows 'Spanish (Spain)'. On the right, there are 'Save Model' and 'Build Model' buttons. The main area is titled 'CUSTOM' and contains sections for 'Interaction Model', 'Invocation', and 'Intents'. Under 'Intents', a list shows 'eventosCiudad' selected (indicated by a blue highlight). This intent has two slots: 'ciudad' (blue circle) and 'fecha' (orange circle). Below the intent list, 'Built-In Intents' are listed: 'AMAZON.CancelIntent', 'AMAZON.HelpIntent', 'AMAZON.StopIntent', and 'AMAZON.NavigateHomeIntent'. A section for 'Slot Types (0)' is at the bottom right. To the right of the intent list, a sidebar titled 'Intents / eventosCiudad' shows sample utterances: 'What might a user say to invoke this intent?', 'un meetup {fecha} en {ciudad}', 'meetups {fecha} en {ciudad}', 'quiero ir a un meetup {fecha} en {ciudad}', 'dime meetups en {ciudad} {fecha}', and 'a qué meetup puedo ir en {ciudad} {fecha}'.

# Alexa Hosted-Skill

Templates en Python.

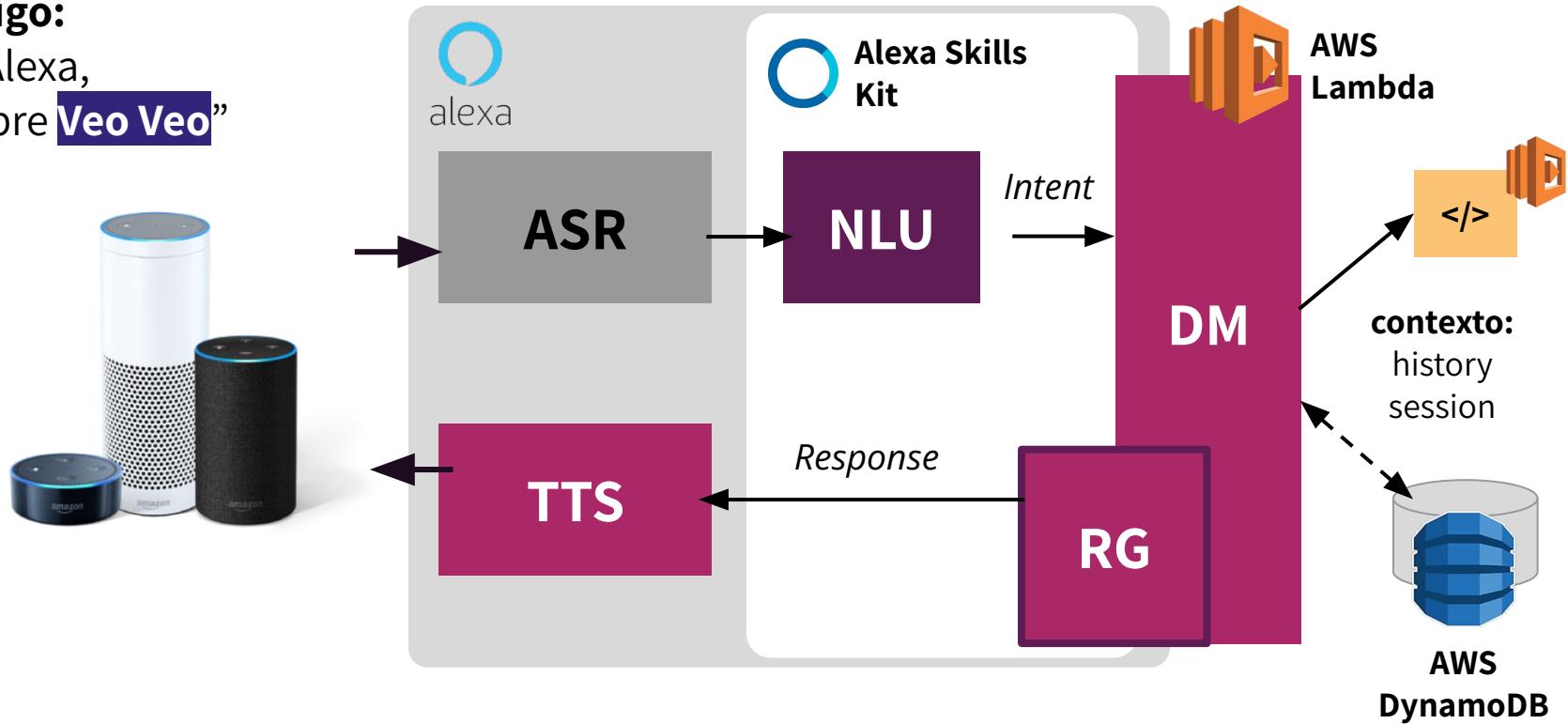
The screenshot shows the Alexa developer console interface. At the top, there's a navigation bar with links like 'Your Skills', 'Pycones', 'Build', 'Code' (which is currently selected), 'Test', 'Distribution', 'Certification', and 'Analytics'. Below the navigation bar is a toolbar with icons for 'New File', 'New Folder', 'Delete', 'Rename', 'DynamoDB Database', 'S3 Storage', 'CloudWatch Logs', 'Usage', 'aws Integrate', 'Download Skill', 'Import Code', 'Offline Tools', and 'Docs'. The main area is a code editor titled 'lambda\_function.py'. The code is written in Python and defines two classes: 'GetNewFactHandler' and 'HelpIntentHandler'. Both classes inherit from 'AbstractRequestHandler'. The 'GetNewFactHandler' class has methods for 'can\_handle' and 'handle', which check if the input is a 'LaunchRequest' or 'GetNewFactIntent'. The 'handle' method retrieves a random fact from a JSON file and creates a speech response. The 'HelpIntentHandler' class also has 'can\_handle' and 'handle' methods, which return a help message. The code editor includes line numbers and syntax highlighting for Python. On the left side of the code editor, there's a sidebar showing the project structure: 'Skill Code' containing 'lambda' (with 'lambda\_function.py', 'language\_strings.json', 'prompts.py', and 'requirements.txt') and other files like 'DynamoDB Database', 'S3 Storage', 'CloudWatch Logs', 'Usage', 'aws Integrate', 'Download Skill', 'Import Code', 'Offline Tools', and 'Docs'. At the bottom of the screen, there are language selection ('English (US)'), feedback ('Feedback'), and copyright information ('© 2010 - 2022, Amazon.com, Inc. or its affiliates').

```
lambda_function.py x
25  class GetNewFactHandler(AbstractRequestHandler):
26      """Handler for Skill Launch and GetNewFact Intent."""
27
28      def can_handle(self, handler_input):
29          # type: (HandlerInput) -> bool
30          return (is_request_type("LaunchRequest")(handler_input) or
31                  is_intent_name("GetNewFactIntent")(handler_input))
32
33      def handle(self, handler_input):
34          # type: (HandlerInput) -> Response
35          logger.info("In GetNewFactHandler")
36
37          # get localization data
38          data = handler_input.attributes_manager.request_attributes["_"]
39
40          random_fact = random.choice(data[prompts.FACTS])
41          speech = data[prompts.GET_FACT_MESSAGE].format(random_fact)
42
43          handler_input.response_builder.speak(speech).set_card(
44              SimpleCard(data[prompts.SKILL_NAME], random_fact))
45          return handler_input.response_builder.response
46
47
48      class HelpIntentHandler(AbstractRequestHandler):
49          """Handler for Help Intent."""
50
51          def can_handle(self, handler_input):
52              # type: (HandlerInput) -> bool
53              return is_intent_name("AMAZON.HelpIntent")(handler_input)
54
55          def handle(self, handler_input):
56              # type: (HandlerInput) -> Response
57              logger.info("In HelpIntentHandler")
58
59              # get localization data
60              data = handler_input.attributes_manager.request_attributes["_"]
61
62              speech = data[prompts.HELP_MESSAGE]
63
64              handler_input.response_builder.speak(speech).set_card(
65                  SimpleCard(data[prompts.SKILL_NAME], speech))
66
67              return handler_input.response_builder.response
```

Digo:

“Alexa,

abre **Veo Veo**”



# Rasa

Crea un chatbot con tecnología con los mejores modelos y algoritmos del estado del arte, y open source.

```
actions.py
# This file contains your custom actions which can
# use custom Python code.
#
# See this guide on how to implement these actions:
# https://rasa.com/docs/rasa/core/actions/#custom-actions

# This is a simple example for a custom action which
# sets the address slot.

from typing import Any, Text, Dict, List

from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher
from rasa_sdk.events import SlotSet

class ActionFacilitySearch(Action):

    def name(self) -> Text:
        return "action_facility_search"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        facility = tracker.get_slot("facility_type")
        address = "300 Hyde St, San Francisco"
        dispatcher.utter_message("Here is the address")

        return [SlotSet("address", address)]
```

```
stories.md
## hospital search
* greet
  - utter_greet
* search_provider{
  "location": "San Francisco"
  - action_facility_search
  - slot_set{"location": "San Francisco"}
* thanks
  - utter_goodbye

## hospital search
* greet
  - utter_greet
* search_provider{
  "location": "San Francisco"
  - utter_ask_location
* inform{"location": "San Francisco"}
  - action_facility_search
* thanks
  - utter_goodbye

## happy path
* greet
  - utter_greet
* mood_great
  - utter_happy

## sad path 1
* greet
  - utter_greet
* mood_unhappy
  - utter_cheer_up
  - utter_did_that_for_you
* affirm
  - utter_happy
```

# RASA Playground (tutorial)

The screenshot shows the RASA Playground tutorial page. The top navigation bar includes the RASA DOCS logo, Rasa, Rasa Enterprise, a search bar, and links for Blog and Community.

The left sidebar contains a navigation menu:

- Introduction
- Rasa Playground** (highlighted)
- Building Assistants
  - Installation
  - Migrate From (beta)
  - Command Line Interface
  - Best Practices >
  - Conversation Patterns >
  - Preparing For Production >
  - Rasa Glossary
- Deploying Assistants >
- Concepts >
- Training Data >
- Domain
- Config >
- Actions >
- Evaluation >
- Channel Connectors >

The main content area is titled "1. NLU data". It contains the following text and code examples:

What are the various things people might say to an assistant that can help them subscribe to a newsletter?

For an assistant to recognize what a user is saying no matter how the user phrases their message, we need to provide example messages the assistant can learn from. We group these examples according to the idea or the goal the message is expressing, which is also called the intent. In the code block on the right, we have added an intent called greet, which contains example messages like "Hi", "Hey", and "good morning".

Intents and their examples are used as training data for the assistant's Natural Language Understanding (NLU) model.

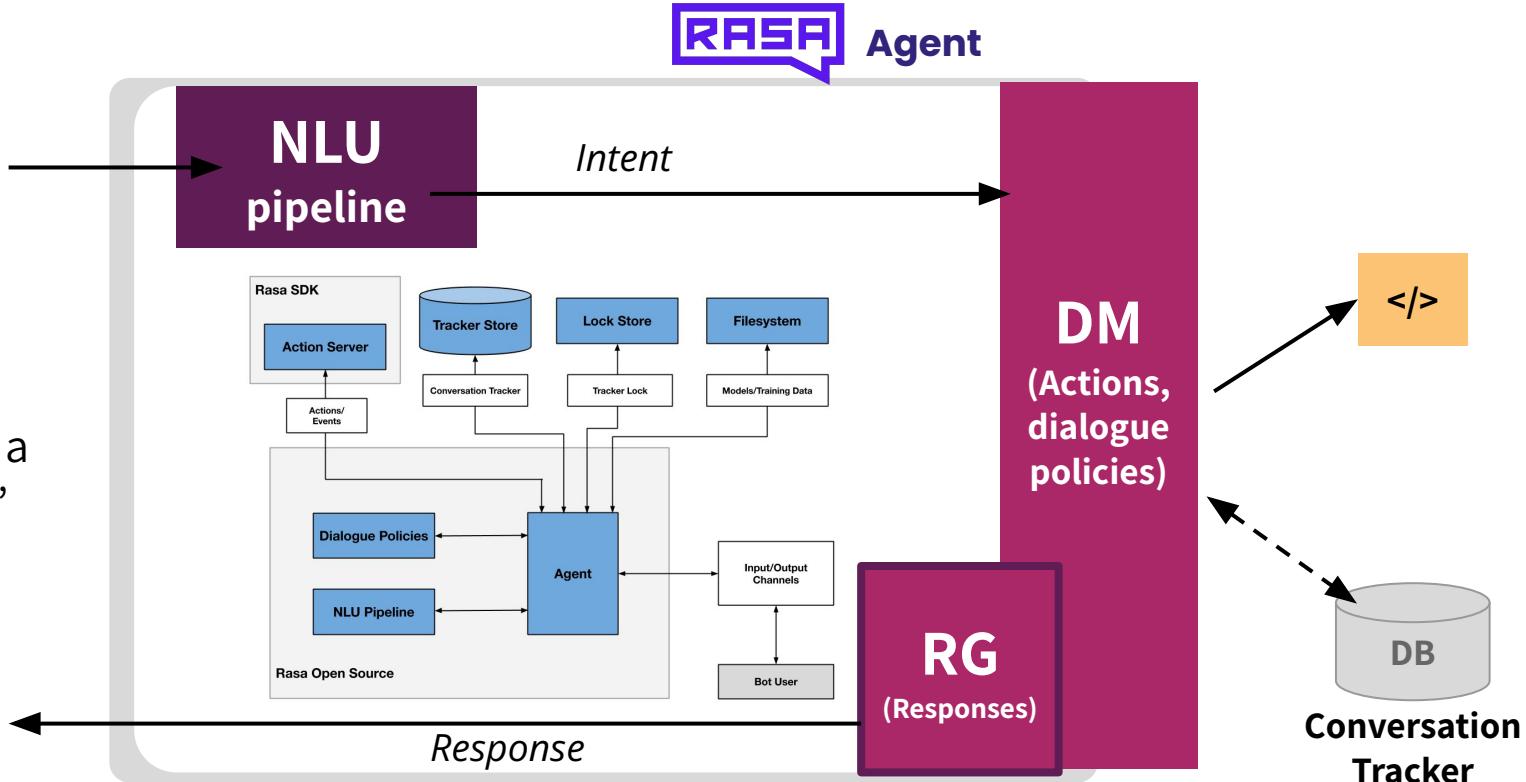
[Learn more about NLU data and its format](#)

```
nlu:  
- intent: greet  
examples: |  
- Hi  
- Hey!  
- Hallo  
- Good day  
- Good morning  
  
- intent: subscribe  
examples: |  
- I want to get the newsletter  
- Can you send me the newsletter?  
- Can you sign me up for the newsletter?  
  
- intent: inform  
examples: |  
- My email is example@example.com  
- random@example.com  
- Please send it to anything@example.com  
- Email is something@example.com
```

[Next step >](#)



**escribo:**  
“quiero ir a  
**Granada**”



**Arquitectura  
end-to-end.**

# GPT-3

## (OpenAI, 2020)

Modelos del  
lenguaje con  
múltiples  
habilidades.

### GPT-3

Our GPT-3 models can understand and generate natural language. We offer four main models with different levels of power suitable for different tasks. Davinci is the most capable model, and Ada is the fastest.

LATEST MODEL	DESCRIPTION	MAX REQUEST	TRAINING DATA
text-davinci-002	Most capable GPT-3 model. Can do any task the other models can do, often with less context. In addition to responding to prompts, also supports <a href="#">inserting</a> completions within text.	4,000 tokens	Up to Jun 2021
text-curie-001	Very capable, but faster and lower cost than Davinci.	2,048 tokens	Up to Oct 2019
text-babbage-001	Capable of straightforward tasks, very fast, and lower cost.	2,048 tokens	Up to Oct 2019
text-ada-001	Capable of very simple tasks, usually the fastest model in the GPT-3 series, and lowest cost.	2,048 tokens	Up to Oct 2019

While Davinci is generally the most capable, the other models can perform certain tasks extremely well with significant speed or [cost advantages](#). For example, Curie can perform many of the same tasks as Davinci, but faster and for 1/10th the cost.

We recommend using Davinci while experimenting since it will yield the best results. Once you've got things working, we encourage trying the other models to see if you can get the same results with lower latency. You may also be able to improve the other models' performance by [fine-tuning](#) them on a specific task.

# Open domain conversations: end-to-end

Save View code Share ...

Tweet:

¿Has probado ya a GPT-3? ¿Qué te parece? Ojos

Si estás haciendo pruebas/experimentos/proyectos con él, me gustaría conocerlos! ¡Añade como respuesta a este tweet tus descubrimientos!

Y si no sabes cómo empezar, empieza aquí

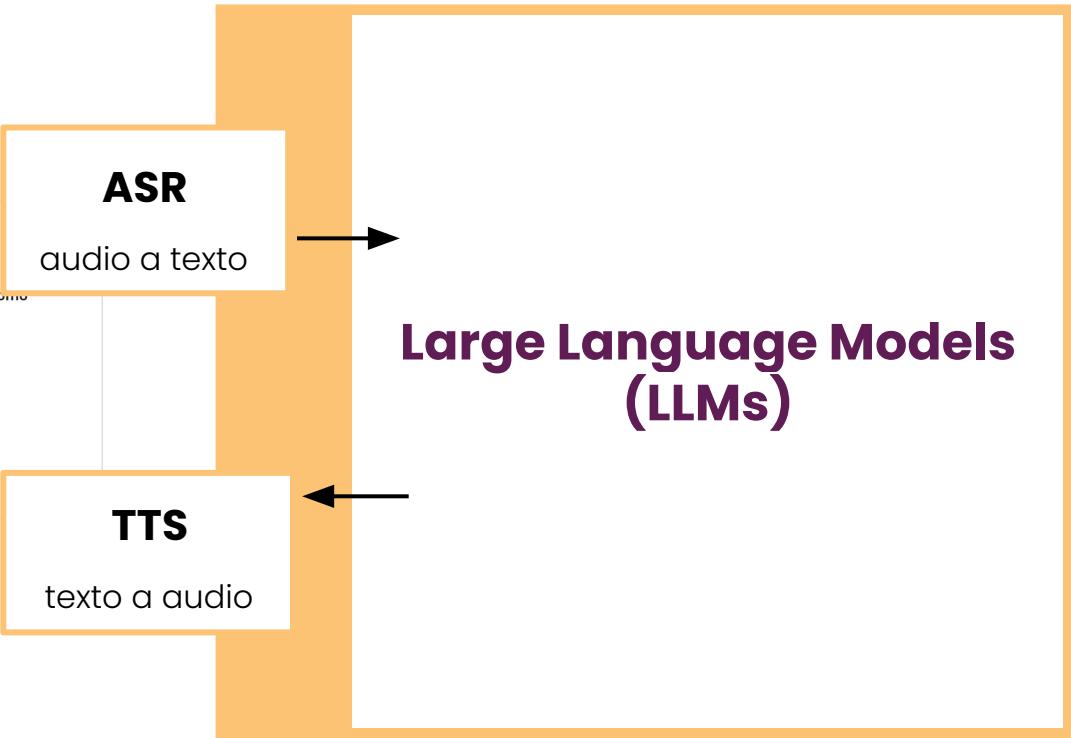
---

Responde como si fueras un hater al tweet anterior:

GPT-3 es una mierda, no sirve para nada.

Generate ⌂ ⌂ ⌂ ⌂

**Credits:** Carlos Santana @DotCSV



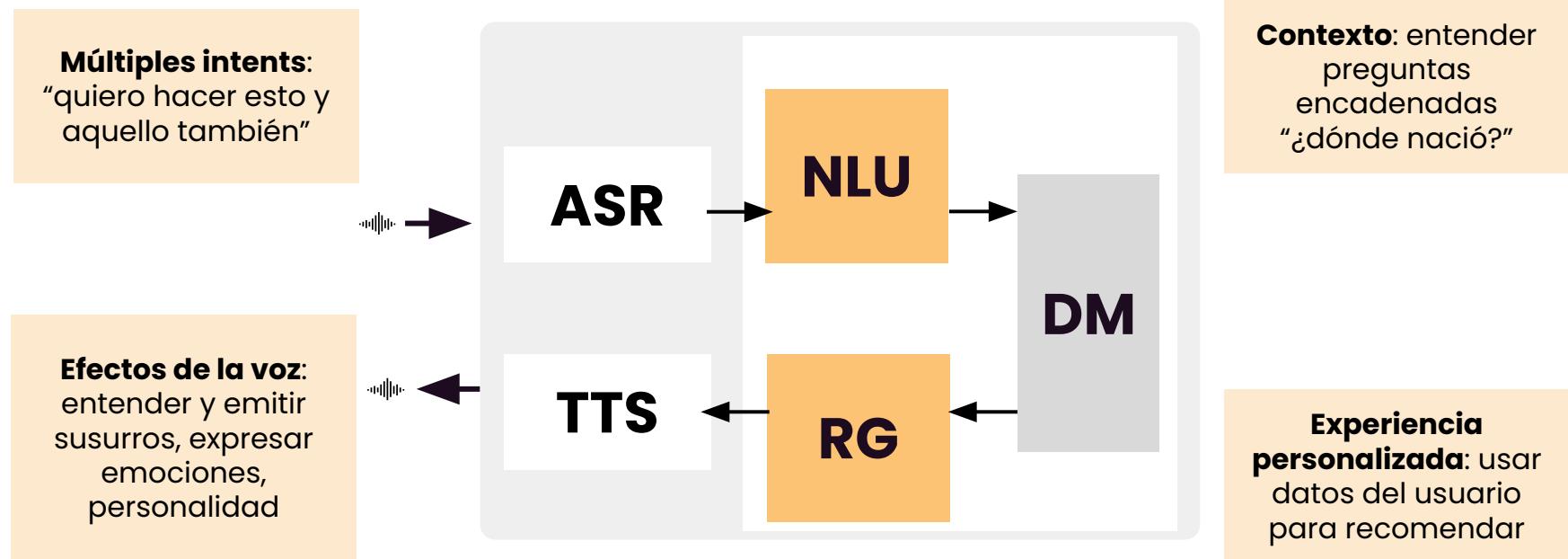


# Y qué más vendrá...

Más estado del arte de la IA  
conversacional.

# Más SOTA (State of the art)

Disponibles en algunos asistentes, o en inglés, que van orientadas a buscar la naturalidad en la conversación.



---

## **En general, estos son los retos de la IA conversacional**

**Entender el  
CONTEXTO y  
mantenerlo**

**APRENDIZAJE**  
Activo y a largo plazo  
Multimodalidad y multitarea

Conversaciones  
**OPEN-DOMAIN**

Estos avances  
tecnológicos son  
apasionantes ❤️

... hay  
cualidades NO humanas  
de la IA conversacional  
que nos dan valor..



... tecnología que nos  
ayuda, que nos hace  
la vida más fácil



no nos distraigamos  
antropomorfizando,  
dando cualidades humanas  
a la tecnología..



**Los aviones son  
probados según su  
habilidad para volar,  
no comparándolos con aves.**

**“IA: un enfoque moderno” – Russell y Norvig**

# ¡Gracias por vuestra atención!



## NIEVES ÁBALOS

Cofundadora y CPO @ Monoceros Labs

✉️ [nieves@monoceroslabs.com](mailto:nieves@monoceroslabs.com)

🐦 [@nieves\\_as](https://twitter.com/nieves_as)

💼 [/in/nievesabalosserrano](https://www.linkedin.com/in/nievesabalosserrano)