

# Natural Language Processing

**NEXTDIGITAL HUB**

Marc Marín Melchor

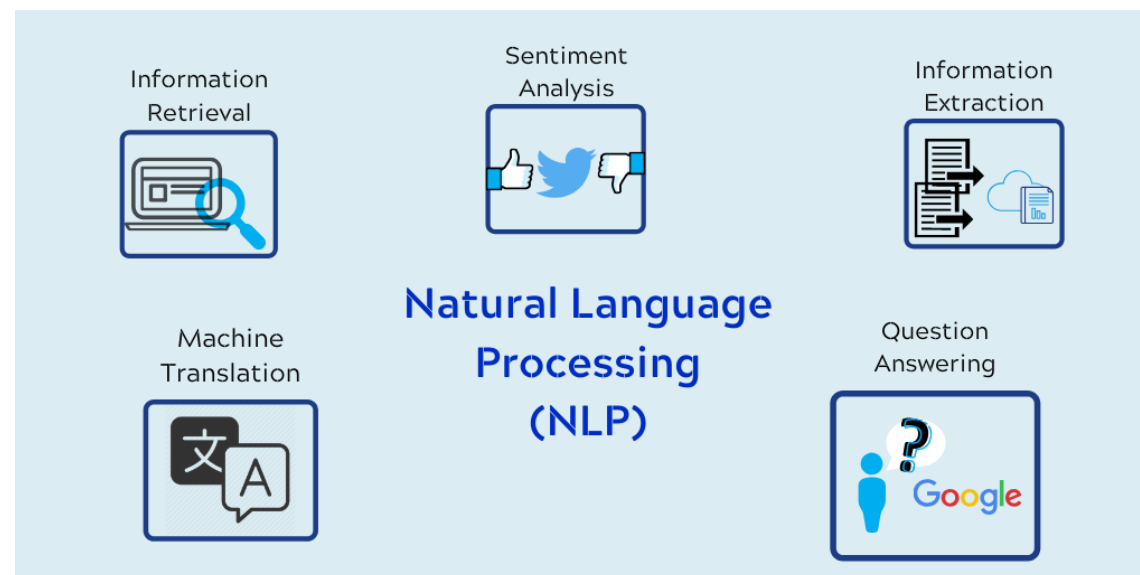
Curso 21/22 - Edición 2

Fecha 26/02/2021

# NLP

## OBJETIVO DE LA CLASE

- Introducción al NLP
- Evolución y Aplicaciones del NLP
- Librerías de Python para NLP (NLTK, Spacy...)
- Técnicas de NLP (BoW, Word Embedding, W2Vec)
- Estado del Arte.



## INTRODUCCIÓN AL NLP

## Introducción al Natural Language Processing

Empecemos por lo básico:



Todos Hablamos



Todos Leemos

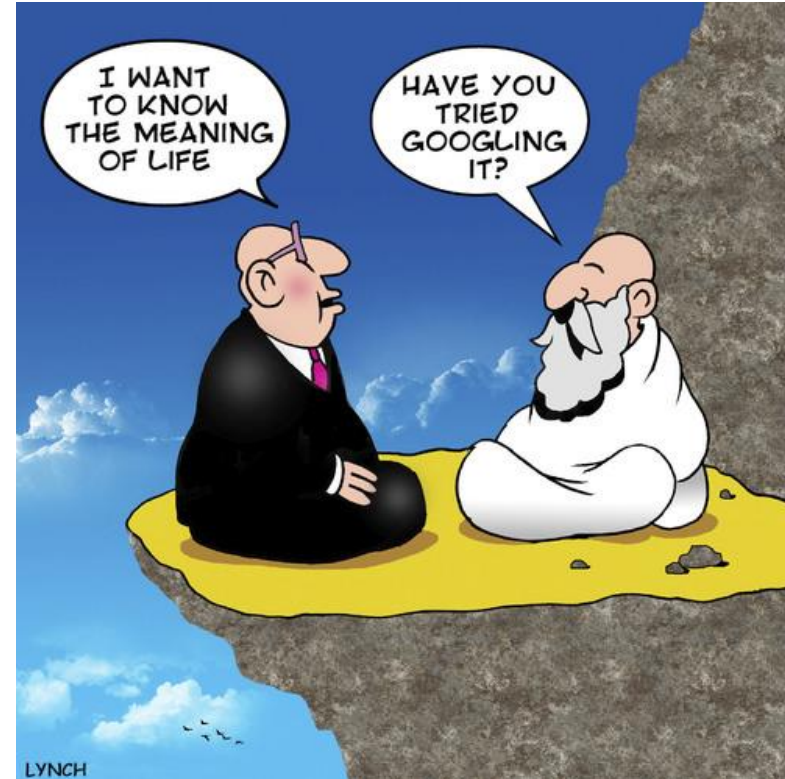


Todos Escribimos

Todos Esto utilizando el lenguaje

## PERO ESTO NO ES TODO...

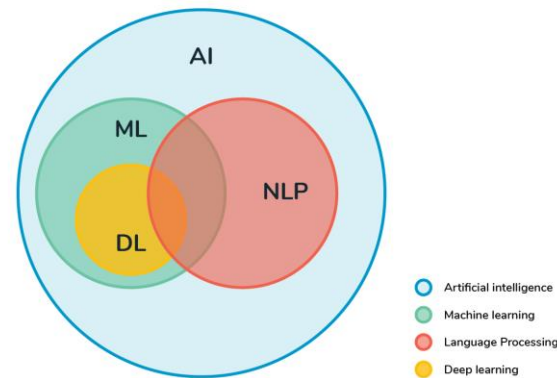
- Pensamos en el mundo que nos rodea
- Soñamos
- Tomamos decisiones
- Elaboramos planes.
- Todo en Lenguaje Natural, utilizando palabras



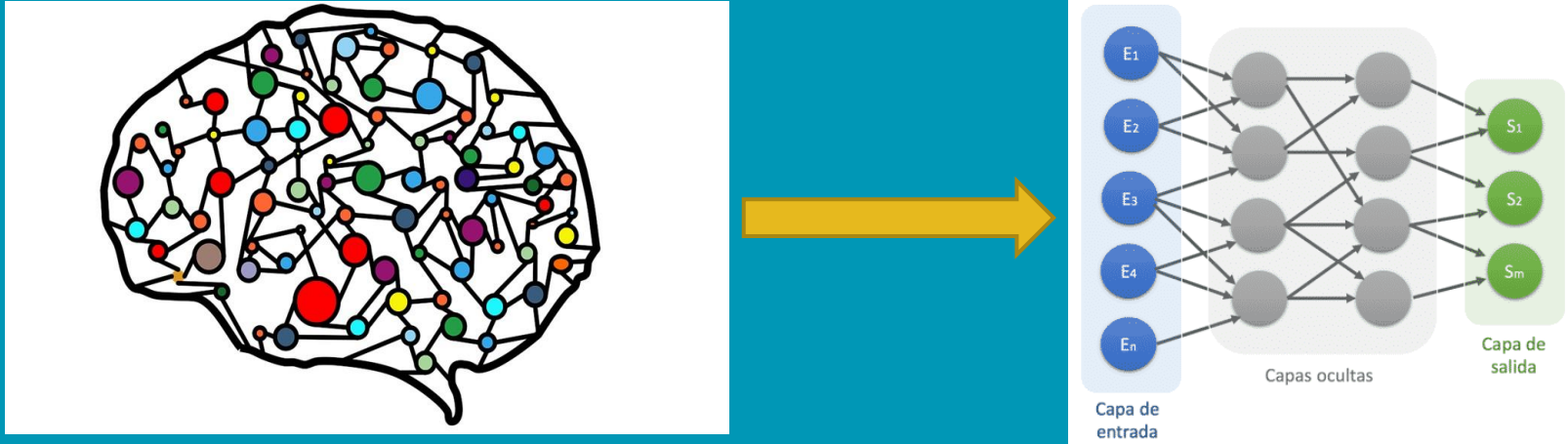
## ¿Qué es Natural Language Processing?

- Campo de estudio centrado en dar sentido al lenguaje
  - Utilizando estadística y los PC

*El procesamiento del lenguaje natural o NLP es un campo de la inteligencia artificial que brinda a las máquinas la capacidad de leer, comprender y derivar el significado de los lenguajes humanos.*



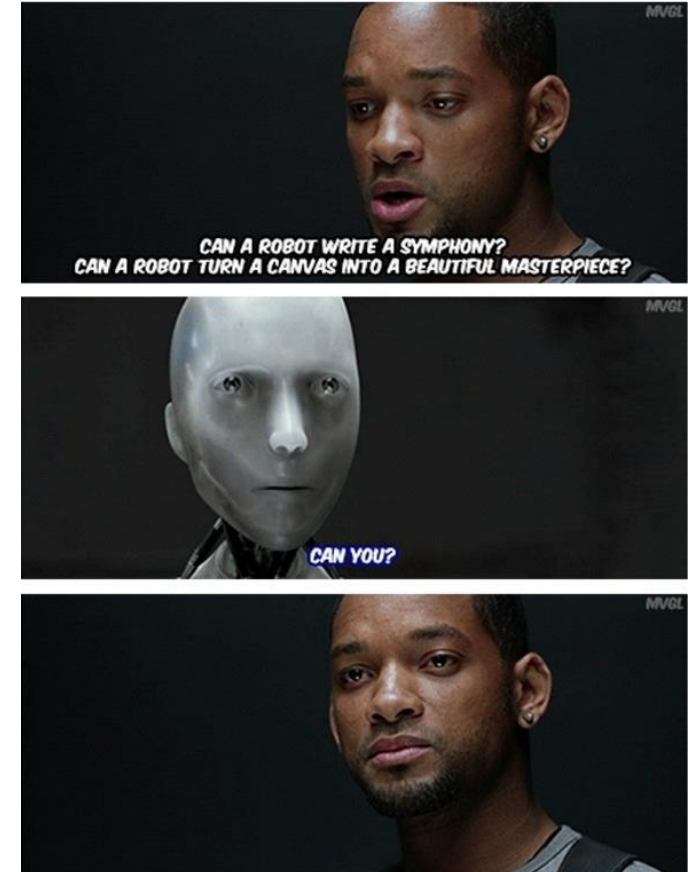
# Redes Neuronales y Deep Learning



- Algoritmos de Redes Neuronales Artificiales están inspirados por las Redes Neuronales Biológicas.
- Intentan modelar las respuestas neuronales en el cerebro asociadas a diversos estímulos.
- Deep Learning se refiere a las ANN con un alto número de capas intermedias.
- Estos algoritmos funcionan bien con datos no etiquetados a gran escala (ej: lenguaje)
- La gran mayoría de los últimos avances en tareas de NLP se deben gracias a los avances en ANN y DL

## NLP, Redes Neuronales y DL

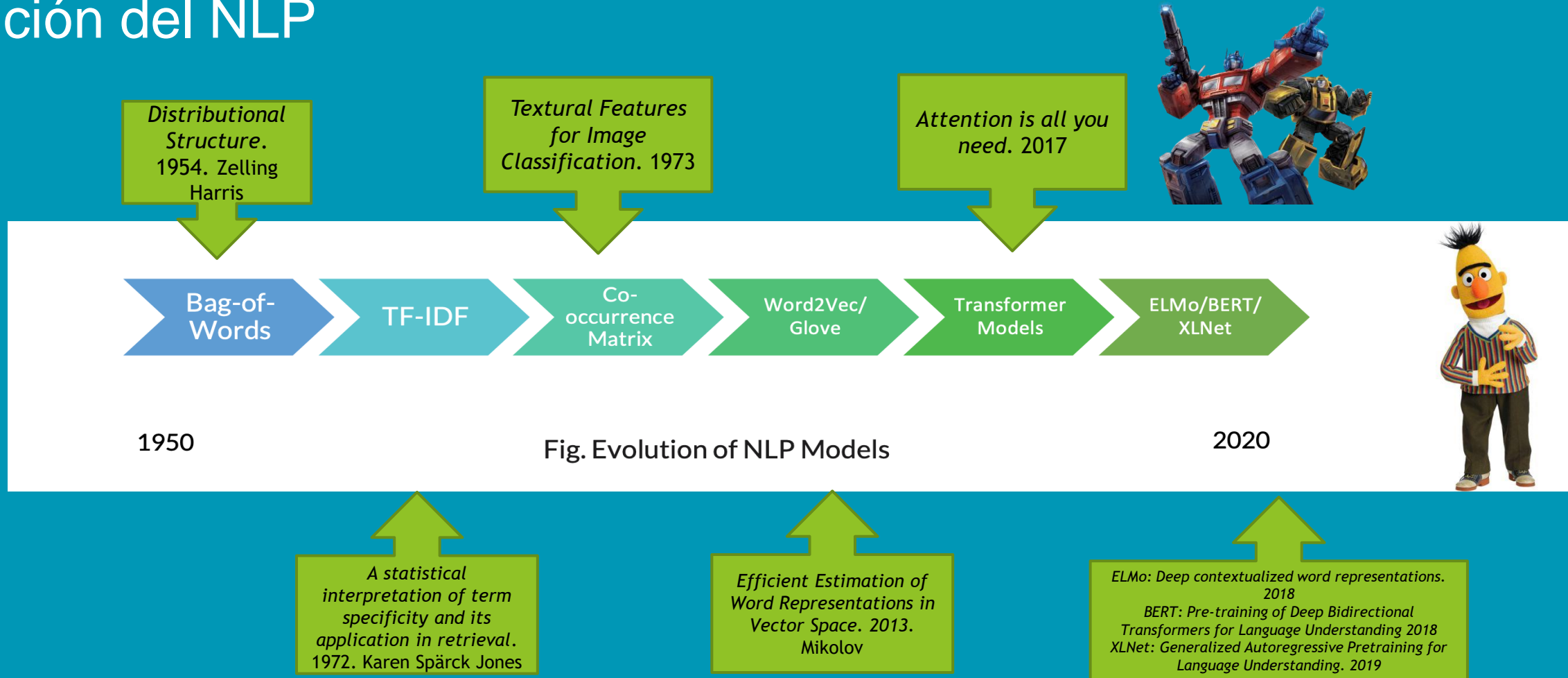
- ANN y DL intentan imitar la forma en que el cerebro humano procesa la información.
- Han permitido grandes hitos en tareas que tradicionalmente eran muy difíciles de resolver con ML:
  - Computer Vision
  - Comprensión y razonamiento lingüístico
- La información que llega no está estructurada. No entendemos bien cómo nuestro cerebro la procesa, cómo resuelve la tarea, qué características tiene.





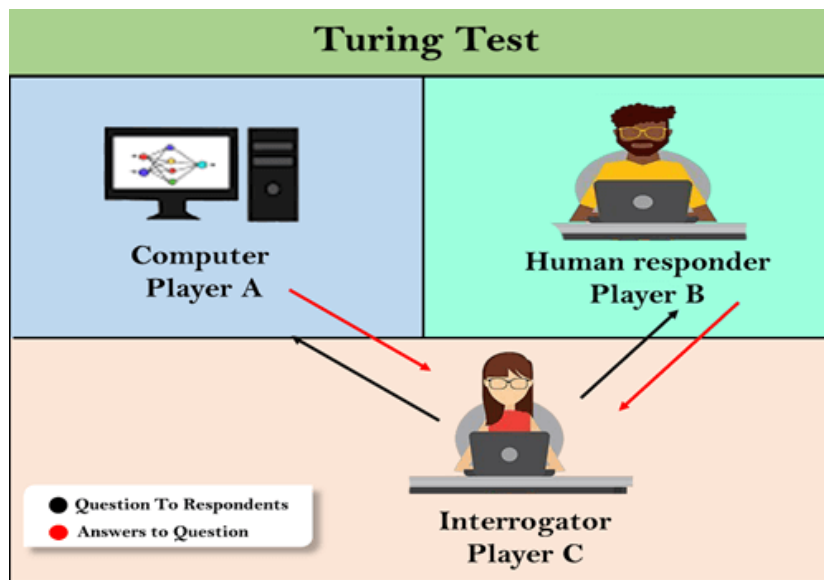
## EVOLUCIÓN Y APLICACIONES DEL NLP

## Evolución del NLP



## Test de Turing.

- En 1950, Alan Turing escribió un Paper llamado “[Computing Machinery And Intelligence](#)”.
- Turing se preguntaba: ¿Pueden las máquinas pensar?
- El **test de Turing** es una herramienta para determinar ‘empíricamente’ si un ordenador ha llegado a desarrollar inteligencia o no.



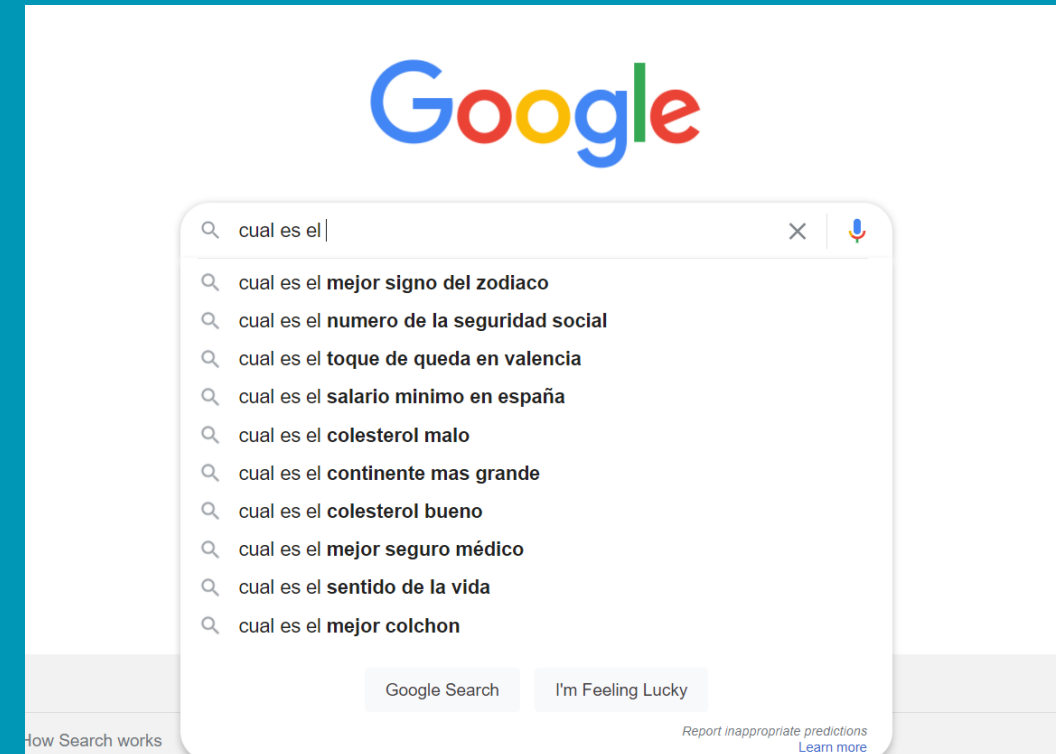
## Aplicaciones del NLP

### 1) Autocorrección de Búsqueda y Autocompletar.

- Se trata de un modelo de lenguaje (LM -language model) en el campo del NLP.
- Estos modelos aprenden a predecir la probabilidad de una secuencia de palabras

Word ordering:

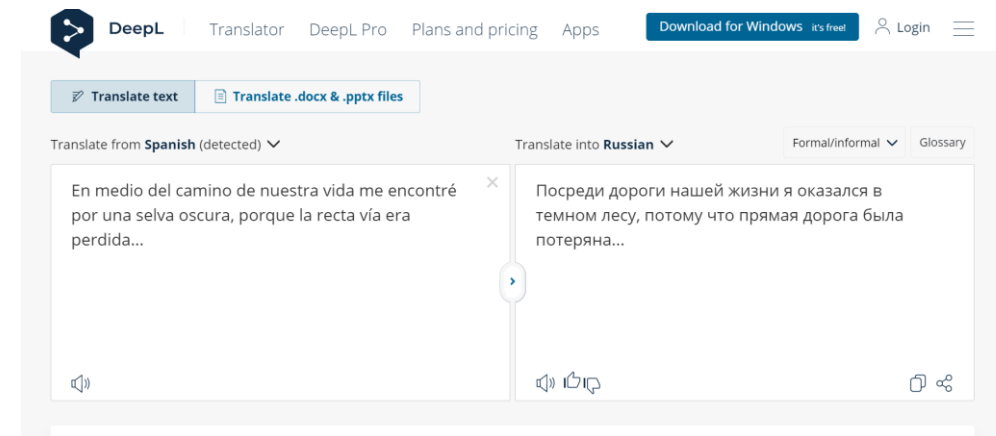
$p(\text{the cat is small}) > p(\text{small the is cat})$



## Aplicaciones del NLP

### 2) Machine Translation

- Machine translation es el procedimiento de convertir automáticamente el texto de un idioma a otro manteniendo el significado intacto.
- Originalmente estaban basados en sistemas de reglas y diccionarios.
- Con la evolución de las redes neuronales, el big data y la potencia computacional, ha dado un salto de calidad.



## Aplicaciones del NLP

### 3) Monitorización de las Redes Sociales.

- Varias técnicas de Sentiment Analysis son utilizada por las compañías para saber qué piensan los clientes sobre sus productos.
- También se utilizan para saber qué problemas afrontar al utilizar esos productos.
- Los gobiernos lo utilizan por temas de seguridad.



## Aplicaciones del NLP

### 3) Chatbots

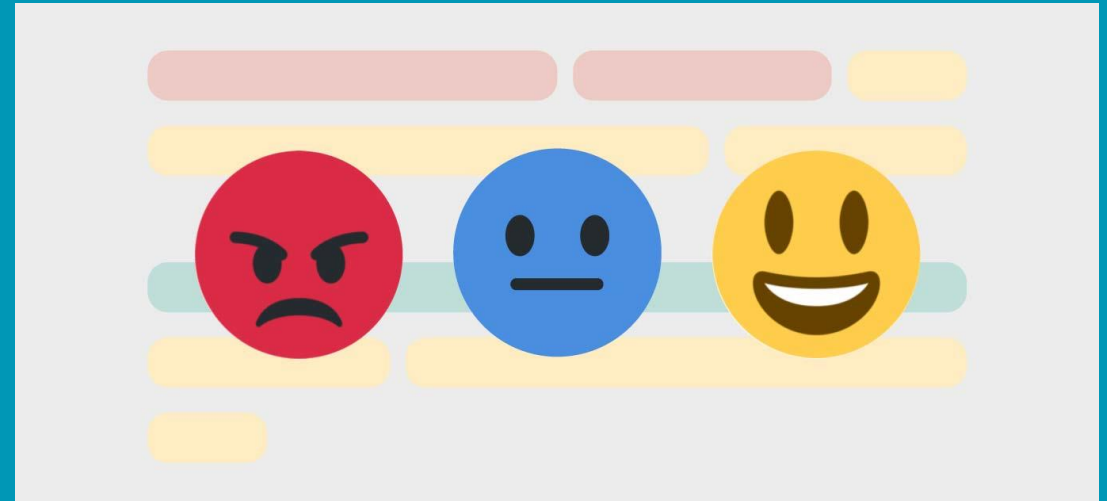
- Ayudan a las empresas a mejorar el producto.
- Miden las necesidades de los consumidores.
- Ayudan a realizar determinadas tareas.
- Básicamente hay 2 variantes:
  - Rule-Based chatbot. Basados en árbol de decisión
  - AI Chatbot. Utilizan modelos más complicados de ML.
- Ejemplo de Chatbot GPT-3



## Aplicaciones del NLP

### 4) Análisis de Encuestas

- Encuestas evalúan el desarrollo de una compañía.
- Gran volumen de datos, entra el NLP





## Aplicaciones del NLP

### 5) Publicidad Dirigida

- Tipo de publicidad en la que se muestran anuncios al usuario en función de su actividad online.
- Gracias al NLP, consiguen que los anuncios vayan únicamente a potenciales clientes.



## Aplicaciones del NLP

### 6) Recruiters y Trabajos

- Hay puestos de trabajo donde hay que analizar miles de CV. NLP Ayuda a automatizar la Tarea.
- Utilizando el Named Entity Recognition se puede extraer información importante como Habilidades, Nombre, Localización y Educación.
- Con este Input se puede clasificar al Candidato como Apto/No Apto



## Aplicaciones del NLP

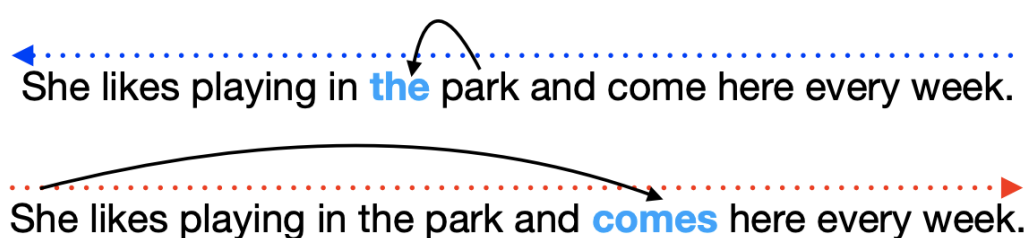
### 7) Correctores gramaticales.

- Una de las aplicaciones más utilizadas en el campo del NLP
- Ayudan a escribir mejor.
- Han mejorado mucho comparado con los antiguos.

She likes playing in **park** and **come** here every week.

She likes playing in **the** park and come here every week.

She likes playing in the park and **comes** here every week.



## Aplicaciones del NLP

### 8) Asistentes Virtuales.

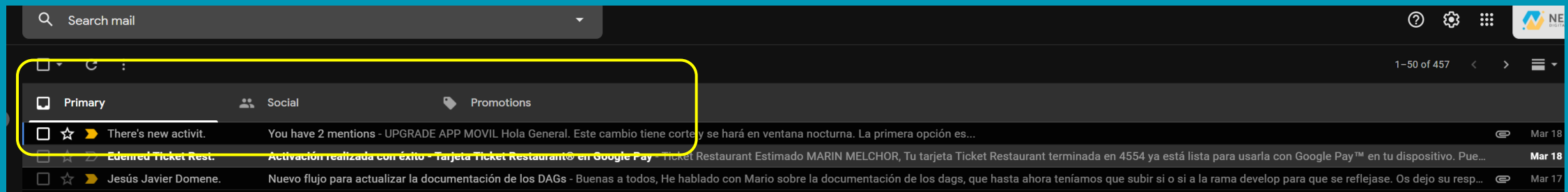
- Es un Software que utiliza Speech Recognition y Natural Language Understanding para entendernos.
- Similar a un chatbot



# Aplicaciones del NLP




## 9) Filtros en los Correos

- Usan algoritmos de clasificación de texto.
- Clasificación de Texto es el proceso de clasificar una parte del texto en alguna de las categorías predefinidas.
- Son independientes de los filtros de SPAM



## LIBRERÍAS Y TÉCNICAS DE NLP

# ¿Qué son Expresiones Regulares?

- Strings con Sintáxis especial  Encontrar todos los enlaces en una web.
- Permiten encontrar patrones en otros strings.  Analizar las direcciones de correo electrónico
- Aplicaciones expresiones regulares  Eliminar/Reemplazar caracteres innecesarios.

```
import re  
re.match('abc', 'abcdef')
```

```
<re.Match object; span=(0, 3), match='abc'>
```

```
word_regex = '\w+'  
re.match(word_regex, 'Hola, qué tal')
```

```
<re.Match object; span=(0, 4), match='Hola'>
```

## Patrones Regex Comunes

Patrón	Match	Ejemplo
\w+	palabra	'Magia'
\d	dígito	9
\s	espacio	' '
.*	wildcard	'username74 '
+ ó *	greedy match	'aaaaaa'
\S	NO espacio	'no_espacio'
[a-z]	grupo minúsculas	'aaaaaa'



## Módulo re

- Módulo `re`
- `split` : dividir un string en regex
- `findall` : encontrar todos los patrones en un string.
- `search` : buscar un patrón
- `match` : coincidir un string o substring basados en un patron
- Pattern (patrón) primero y el string segundo.
- Puede devolver un iterador, string u objeto match.

```
re.split('\s', 'Divide los Espacios')
```

```
['Divide', 'los', 'Espacios']
```

## ¿Qué es Tokenización?

- Transformar un string o documento en **Tokens** (**chunks** más pequeños)
- Un paso en la preparación de texto para NLP
- Muchas teorías y reglas
- Podemos crear nuestras propias reglas utilizando expresiones regulares
- Algunos ejemplos
  - Dividir palabras u oraciones
  - Separar partes de una cadena
  - Separar todos los hashtags en un tweet.



## NLTK Librería

- *nltk*: Natural Language Toolkit

```
from nltk.tokenize import word_tokenize  
word_tokenize("Hi there!")
```

```
['Hi', 'there', '!']
```

## ¿Por qué Tokenizar?

- ☐ Más fácil de mapear parte de las frases
- ☐ Encontrar palabras comunes
- ☐ Eliminar palabras no deseadas
- ☐ *"Sra. kidley, no me gustan sus zapatos... su vestido... ni su sombrero."*
- ☐ " Sra" , " Kidley", "no", "me", "gustan", "sus", "zapatos", "su", "vestido", "ni" , "su", "sombrero"

## Otros tokenizers de nltk

- **sent\_tokenize** : tokeniza un documento en frases.
- **regexp\_tokenize**: tokeniza un string o documento, basado en un patrón de expresiones regulares.
- **TweetTokenizer**: clase especial que se usa solamente para la tokenización de tuits, permitiendo separar #hashtags, @menciones y muchísimos símbolos de exclamación!!!



## Más práctica Regex

- Diferencias entre `re.search()` y `re.match()`

```
import re
re.match('abc', 'abcde')
```

```
<_sre.SRE_Match object; span=(0, 3), match='abc'>
```

```
re.search('abc', 'abcde')
```

```
<_sre.SRE_Match object; span=(0, 3), match='abc'>
```

```
re.match('cd', 'abcde')
re.search('cd', 'abcde')
```

```
<_sre.SRE_Match object; span=(2, 4), match='cd'>
```

~~Match no encuentra coincidencias~~

## Grupos Regex usando or “|”

- OR está representado por ‘el carácter |’
- Podemos definir un grupo utilizando ()
- También podemos definir un rango explícito de caracteres utilizando []

```
import re
match_digits_and_words = ('(\d+|\w+)')
re.findall(match_digits_and_words, 'He has 11 cats.')
```

## Grupos y Rangos en Regex

Pattern	Match	Ejemplo
<code>[A-Za-z]+</code>	Mayúsculas y minúsculas alfabeto Inglés	<code>'ABCdefGHIjK'</code>
<code>[A-Za-zñÑ]+</code>	Mayúsculas y minúsculas alfabeto Español	<code>'LMnÑñOPQrstuV'</code>
<code>[A-Za-z\-\.\.]+</code>	Mayúsculas, minúsculas, - y .	<code>'me-encanta-python.com'</code>
<code>(a-z)</code>	a, - y z	<code>'az-'</code>
<code>(\s+l,)</code>	Espacios o una coma	<code>' , '</code>



## Rango de caracteres con `re.match()`

```
import re  
my_str = 'match lowercase spaces nums like 12, but no commas'  
re.match('[a-z0-9 ]+', my_str)
```

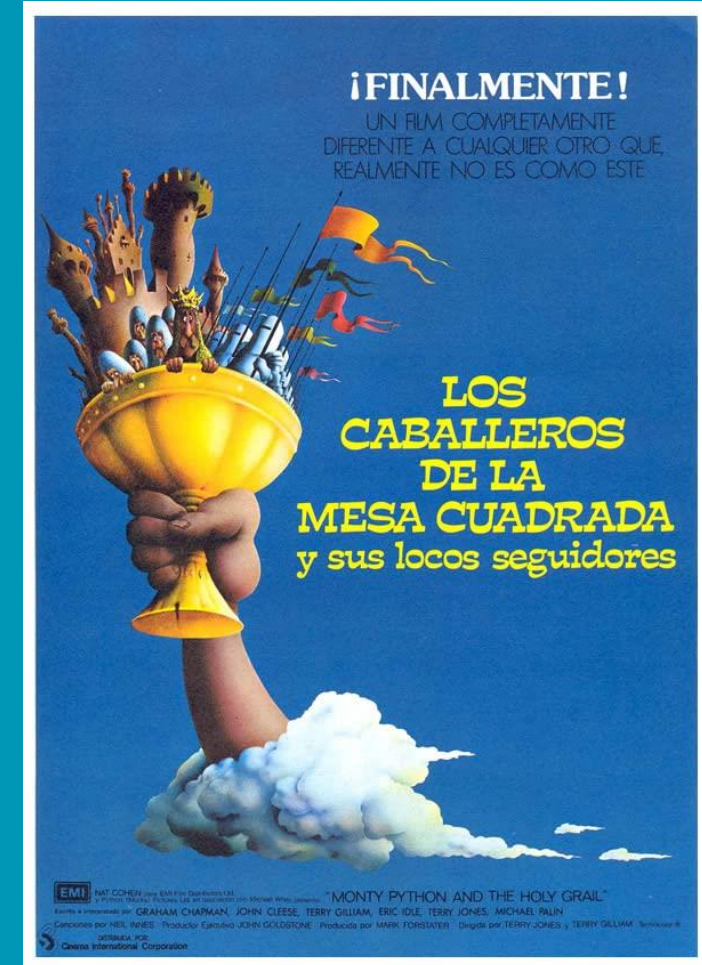
```
<_sre.SRE_Match object;  
  span=(0, 42), match='match lowercase spaces nums like 12'>
```

## Bag of Words BoW

- Método básico para encontrar tópicos en un texto
- Necesitamos primero crear tokens usando la tokenización.
- ... y entonces contar todos los tokens que tiene.
- A mayor frecuencia de una palabra, más importante puede ser.
- Puede ser una forma excelente de determinar las palabras significativas en un texto según la cantidad de veces que se usan.

## Ej BoW – Reviews Cine

1. *"La película más divertida del 75 y probablemente también es la película más tonta."*
2. *Clásico Monty Python es la mejor!! ¡Es tan divertida!*
3. *No es graciosa en absoluto y es muy aburrida.*



## Ej BoW - Reviews Cine

- No debería dar lugar a una matriz dispersa. Tienen un coste elevado.
- Queremos retener la mayor parte de información

	película	más	divertid a	en	es	graciosa	mejor	monty	absoluto	[...]
Review 1	1	1	1	0	1	0	0	0	0	...
Review 2	0	0	1	0	1	0	1	1	0	...
Review 3	0	0	0	1	1	1	0	0	1	...

```
[141] fdist
```

```
{'75': 1,
 'absoluto': 1,
 'aburrida': 1,
 'clásico': 1,
 'del': 1,
 'divertida': 2,
 'en': 1,
 'es': 4,
 'graciosa': 1,
 'la': 3,
 'mejor': 1,
 'monty': 1,
 'muy': 1,
 'más': 2,
 'no': 1,
 'película': 2,
 'probablemente': 1,
 'python': 1,
 'también': 1,
 'tan': 1,
 'tonta': 1}
```

# PREPROCESAMIENTO DE TEXTO

- Ayuda a mejorar los datos de entrada
  - Cuando realizamos ML u otros métodos estadísticos.
- Ejemplos:
  - Tokenización para crear el BoW
  - Convertir a minúsculas
- Lematización/*Stemming*
  - Acortar palabras a la raíz (lexema)
- Eliminando Stop Words, puntuación o tokens no deseados

## Lematización - Stemming

**Stemming** - Estos algoritmos funcionan **cortando el final** (o el principio) de una palabra, teniendo en cuenta una lista de prefijos y sufijos comunes que podemos encontrar en una palabra flexionada. Presenta algunas limitaciones.

Form	Suffix	Stem
stud <b>ies</b>	-es	studi
stud <b>ying</b>	-ing	study
niñ <b>as</b>	-as	niñ
niñ <b>ez</b>	-ez	niñ

**Lematización** - tiene en cuenta el **análisis morfológico** de las palabras. Para ello, es necesario disponer de diccionarios detallados que el algoritmo puede consultar para relacionar la forma con su lema.

- Importante: un Lema es la forma base de todas sus formas flexivas, mientras que una raíz no lo es.

Form	Morphological information	Lemma
studies	Third person, singular number, present tense of the verb <b>study</b>	study
studying	Gerund of the verb <b>study</b>	study
niñas	Feminine gender, plural number of the noun <b>niño</b>	niño
niñez	Singular number of the noun <b>niñez</b>	niñez

## Stop Words

- Son palabras que por si misma no significan nada, si no que acompañan a otras.
- En español, este grupo suele estar conformado por **artículos, pronombres, preposiciones, adverbios y algunos verbos** .
- Además de en NLP, Son muy usadas en los buscadores y en temas de SEO.

Stop Words - Español	Stop Words - Inglés
a	me
el	my
entonces	we
os	our
te	you
ti	by
[...]	at

## Preprocesamiento de Texto - Ejemplo

Texto de Entrada: *“NLP, Machine Learning and neural networks are common fields of AI. So are robotics.”*



Tokens de Salida: nlp, machine, learning, neural, network, common, field, ai, robotics.



# Preprocesamiento en Python

```
from nltk.corpus import stopwords
text = """The cat is in the box. The cat likes the box.
        The box is over the cat."""
tokens = [w for w in word_tokenize(text.lower())
          if w.isalpha()]
no_stops = [t for t in tokens
            if t not in stopwords.words('english')]
Counter(no_stops).most_common(2)
```

```
[('cat', 3), ('box', 3)]
```

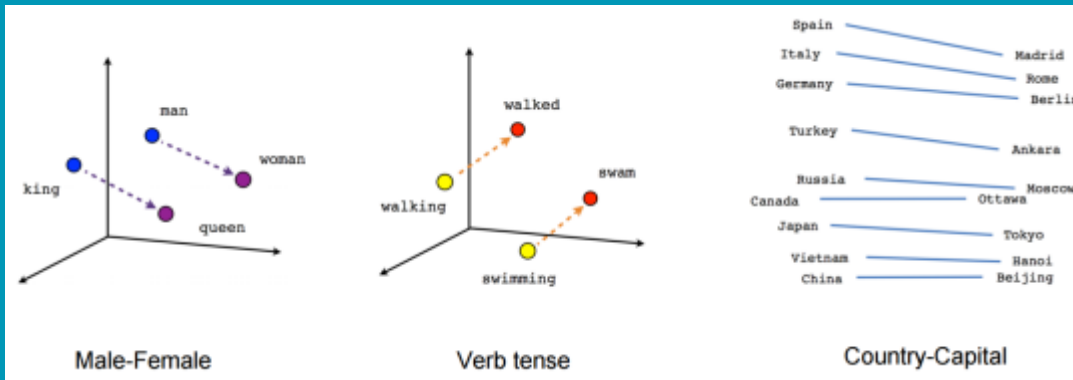
# NLP

## Word 2 Vec

PRACTICAS!!!!

# Word Embedding

- Un Word Embedding o Word Vector es la representación numérica de una palabra en forma de vector.
- Cada palabra en el vocabulario, tendrá un vector único asociado.
- Las palabras más cercanas en significado, irán más cercanas en el vector de representación (por la distribución de probabilidad).



## Word Embedding - Example

### CORPUS

It is a nice day out

Is it nice out?

Nice Day

It is a **nice day** out

Is it nice out?

Nice Day



	it	is	a	nice	day	out
it	0					
is	2	0				
a	1	1	0			
nice	2	2	1	0		
day	1	1	1	2	0	
out	2	2	1	2	1	0

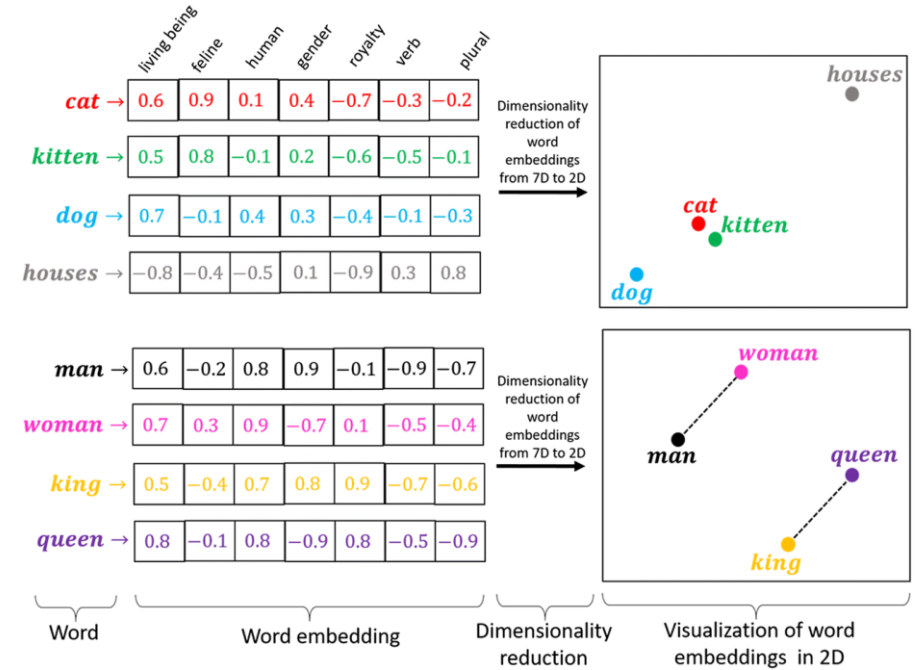
## Word Embedding - Example

- Los vectores de cada palabra se inician de forma aleatoria
- Se actualizan los valores para que los vectores de las palabras que ocurren juntas con más frecuencia estén más cerca unos de otros
- **GloVe = Global Vectors**
- Otras implementaciones conocidas son:
  - FastText
  - Word2Vec

	it	is	a	nice	day	out
it	0					
is	2	0				
a	1	1	0			
nice	2	2	1	0		
day	1	1	1	2	0	
out	2	2	1	2	1	0

## Word Embedding. ¿Dónde se usa?

- Embeddings se usan como input a un modelo de ML
- Visualizar patrones de uso en el corpus de entrenamiento.

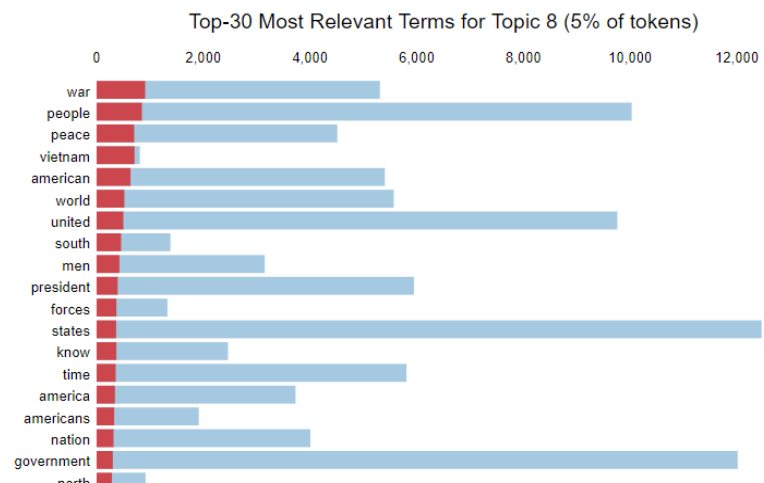


## Librería Gensim

- Librería open-source muy popular para NLP
- Utiliza los mejores modelos académicos para realizar tareas complejas
  - Construir document vectors o word vectors
  - Realizar identificación de tópicos y comparaciones de documentos

# NLP

## Ejemplo de Gensim



LDA – Latent Dirichlet Allocation.


<http://tlfvincent.github.io/2015/10/23/presidential-speech-topics>



## Librería Gensim

```
from gensim.corpora.dictionary import Dictionary
from nltk.tokenize import word_tokenize
my_documents = ['The movie was about a spaceship and aliens.',
                'I really liked the movie!',
                'Awesome action scenes, but boring characters.',
                'The movie was awful! I hate alien films.',
                'Space is cool! I liked the movie.',
                'More space films, please!'],]
```

```
tokenized_docs = [word_tokenize(doc.lower())
                  for doc in my_documents]
dictionary = Dictionary(tokenized_docs)
dictionary.token2id
```




```
{'!': 11,
 ',': 17,
 '.': 7,
 'a': 2,
 'about': 4,
 ...}
```

## Creando un Corpus con Gensim

```
corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]  
corpus
```

```
[[ (0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1)],  
  [(0, 1), (1, 1), (9, 1), (10, 1), (11, 1), (12, 1)],  
  ...]
```

- 
- Modelos *Gensim* pueden ser fácilmente guardados, actualizados y reutilizados.
  - Diccionario puede ser también actualizado

## TF-IDF

- TF-IDF = Term Frequency - Inverse Document Frequency.
- Permite determinar las palabras más importantes en cada documento
- Cada corpus puede tener más palabras compartidas que solo palabras vacías.
- Estas palabras deberían reducir el peso en la importancia.
- Ejemplo de astronomía: 'Cielo'
- Asegura que las palabras más comunes no aparezcan como 'keywords'
- Mantiene las palabras específicas del documento ponderadas alto.

## FÓRMULA TF-IDF

$$w_{x,y} = \text{tf}_{x,y} \times \log \left( \frac{N}{\text{df}_x} \right)$$

### TF-IDF

Term  $x$  within document  $y$

$\text{tf}_{x,y}$  = frequency of  $x$  in  $y$

$\text{df}_x$  = number of documents containing  $x$

$N$  = total number of documents

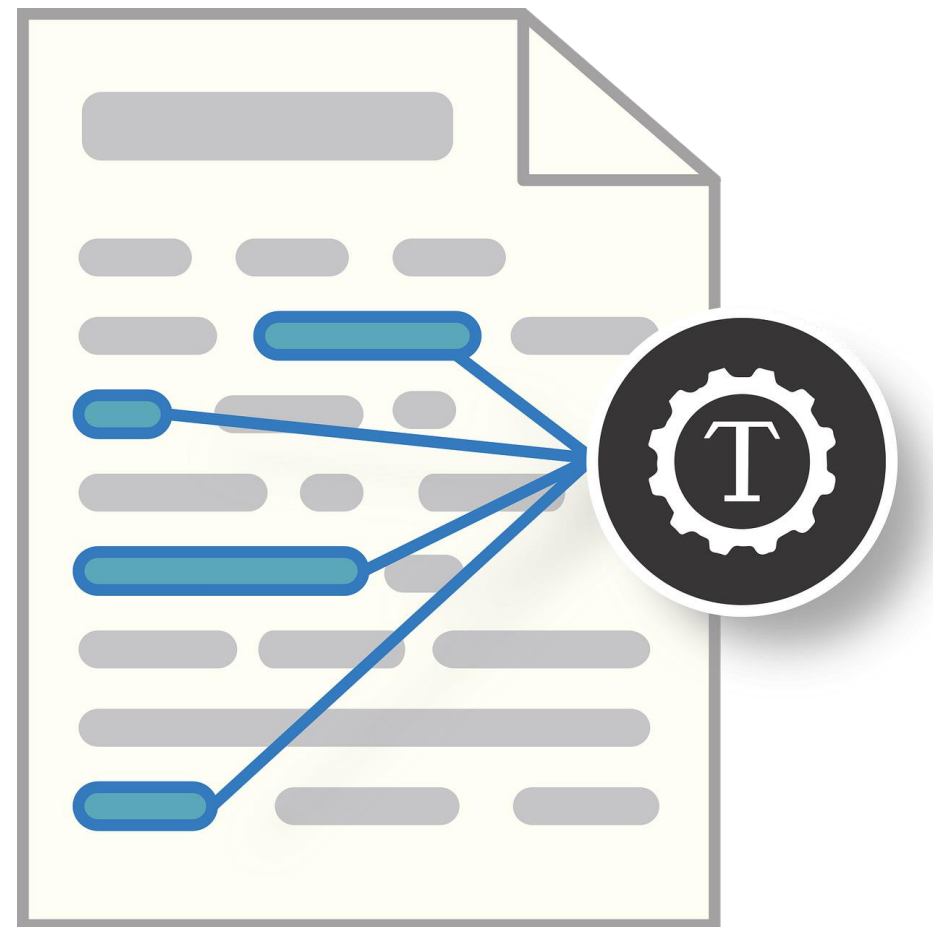
## TF-IDF en gensim

```
from gensim.models.tfidfmodel import TfidfModel  
tfidf = TfidfModel(corpus)  
tfidf[corpus[1]]
```

```
[(0, 0.1746298276735174),  
 (1, 0.1746298276735174),  
 (9, 0.29853166221463673),  
 (10, 0.7716931521027908),  
 ...  
]
```

## Name Entity Recognition - NER

- Es una tarea de NLP utilizada para identificar **entidades nombradas** (NE) en documentos. Algunos ejemplos son:
  - Personas, lugares, organizaciones
  - Fechas, ciudades, obras de arte
  - Marcas, productos, etc.,
- Puede utilizarse junto con Topic identification o por si sola.
- ¿Quién? ¿Qué? ¿Cuándo? ¿Dónde?



## Ejemplo de NER

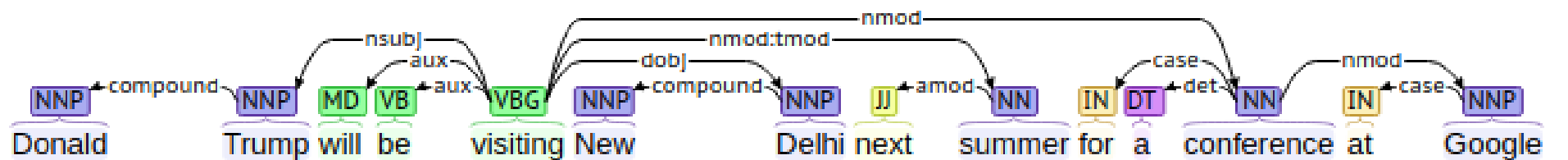
In fact, the **Chinese** **NORP** market has the **three** **CARDINAL** most influential names of the retail and tech space – **Alibaba** **GPE** , **Baidu** **ORG** , and **Tencent** **PERSON** (collectively touted as **BAT** **ORG** ), and is betting big in the global **AI** **GPE** in retail industry space . The **three** **CARDINAL** giants which are claimed to have a cut-throat competition with the **U.S.** **GPE** (in terms of resources and capital) are positioning themselves to become the ‘future **AI** **PERSON** platforms’. The trio is also expanding in other **Asian** **NORP** countries and investing heavily in the **U.S.** **GPE** based **AI** **GPE** startups to leverage the power of **AI** **GPE** . Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing **one** **CARDINAL** , with an anticipated **CAGR** **PERSON** of **45%** **PERCENT** over **2018 - 2024** **DATE** .

To further elaborate on the geographical trends, **North America** **LOC** has procured **more than 50%** **PERCENT** of the global share in **2017** **DATE** and has been leading the regional landscape of **AI** **GPE** in the retail market. The **U.S.** **GPE** has a significant credit in the regional trends with **over 65%** **PERCENT** of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as **Google** **ORG** , **IBM** **ORG** , and **Microsoft** **ORG** .

<https://www.readitquik.com/articles/ai/how-much-will-the-retail-sector-benefit-from-artificial-intelligence/#>

## Nltk y la librería Stanford CoreNLP

- La librería Stanford CoreNLP:
  - Integrada con python vía nltk
  - Basada en Java.
  - Soporte para NER así como coreference y árboles de dependencia





## Introducción a Spacy

- Es una librería de NLP similar a Gensim, con diferentes implementaciones.
- Centrada en crear pipelines de NLP para generar modelos y corpora.
- Open-Source, con librerías y herramientas extra.
  - Displacy

## Displacy Entity Recognition Visualizer

When **Sebastian Thrun** PERSON started working on self-driving cars at **Google** ORG in **2007** DATE, few people outside of the company took him seriously. “I can tell you very senior CEOs of major **American** NORP car companies would shake my hand and turn away because I wasn’t worth talking to,” said **Thrun** ORG, now the co-founder and CEO of online higher education startup Udacity, in an interview with Recode **earlier this week** DATE.

<https://explosion.ai/demos/displacy-ent/>

## ¿Por qué Spacy para NER?

- Creación sencilla de Pipelines
- Diferentes tipos de entidades comparado con NLTK
- Corpus lenguaje informal
- Continua mejora

```
In [62]: text = "When Sebastian Thrun started working on self-driving cars at Google in 2007, \
few people outside of the company took him seriously."
doc = nlp(text)
displacy.render(doc, style="ent", jupyter=True)
```

When Sebastian **NORP** Thrun started working on self-driving cars at Google **ORG** in 2007 **DATE**, few people outside of the company took him seriously.

```
In [46]: doc = nlp(u'Over the last quarter Apple sold nearly 20 thousand iPods for a profit of $6 million. '
u'By contrast, Sony sold only 7 thousand Walkman music players.')
displacy.render(doc, style='ent', jupyter=True)
```

Over the last quarter **DATE** Apple **ORG** sold nearly 20 thousand **CARDINAL** iPods **PRODUCT** for a profit of \$6 million **MONEY**. By contrast, Sony **ORG** sold only 7 thousand **CARDINAL** Walkman music players.

## NER Multi-idioma con Polyglot

- Polyglot es una librería NLP que utiliza Word-Vectors.
- ¿Por qué Polyglot?
  - Vectores para muchos idiomas
  - Word Embeddings más de 130 idiomas
  - Detecta casi 200 idiomas
  - NER en 40 idiomas

乃生男子  
載寢之床  
載衣之裳  
載弄之璋  
其泣啍啍  
朱芾斯皇  
室家君王

When a son is born,  
He is cradled in the bed,  
He is clothed in robes,  
Given a jade sceptre as toy.  
His lusty cries portend his vigor,  
He shall wear bright, red knee-caps,  
Shall be the lord of a hereditary house.

乃生女子  
載寢之地  
載衣之裼  
載弄之瓦  
無非無儀  
唯酒食是議  
無父母詒罹

When a daughter is born,  
She is cradled on the floor,  
She is clothed in swaddling-bands,  
Given a loom-whorl as toy,  
She shall wear no badges of honor,  
Shall only take care of food and drink,  
And not cause trouble to her parents.

# NER en Español con Polyglot

```
from polyglot.text import Text
```

```
text = """El vicepresidente segundo del gobierno y líder de Podemos, Pablo Iglesias, dejará su cargo en el Gobierno para ser candidato a las elecciones de la Comunidad de Madrid del 4 de mayo. Según ha avanzado él mismo este lunes, ha propuesto a Sánchez disgregar la cartera para que la ministra de Trabajo, Yolanda Díaz, asuma la vicepresidencia mientras que Ione Belarra, actual secretaria de Estado de Agenda 2030, se haga cargo del Ministerio de Derechos Sociales. "Madrid está en estos momentos ante un nuevo riesgo, que es un riesgo para Madrid pero también para España; que haya un Gobierno de la ultraderecha", ha advertido Iglesias al anunciar su decisión, que ha compartido en Twitter a través de un video comunicado. """
```

```
ptext = Text(text)
```

```
ptext.entities
```

```
[I-PER(['Pablo', 'Iglesias']),  
I-LOC(['Comunidad', 'de', 'Madrid']),  
I-PER(['Sánchez', 'disgregar']),  
I-PER(['Yolanda', 'Díaz']),  
I-PER(['Ione', 'Belarra']),  
I-ORG(['Sociales']),  
I-LOC(['Madrid']),  
I-LOC(['Madrid']),  
I-LOC(['España']),  
I-PER(['Iglesias'])]
```

## LSTM - Long Short-Term Memory

- LSTM Son un Tipo de Redes Neuronales Recurrentes.
- Eran el estado del arte hasta que llegaron los Transformers.
- RNN son redes neuronales que son buenas para modelar **datos secuenciales**.



## RNN

Las RNN son buenas procesando datos secuenciales para predicciones

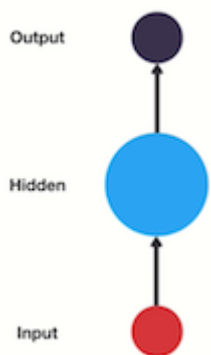
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Z Y X W V U T S R Q P O N M L K J I H G F E D C B A

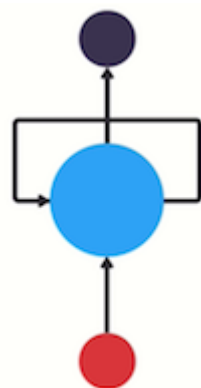
A esto se le llama Memoria Secuencial!

## LSTM - RNN

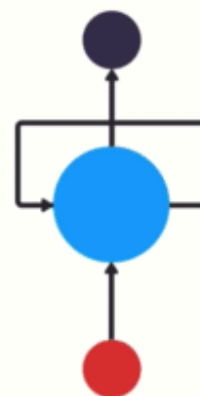
Las RNN tienen este tipo de **Memoria Secuencial**. ¿Cómo lo consiguen?



Feed-Forward NN



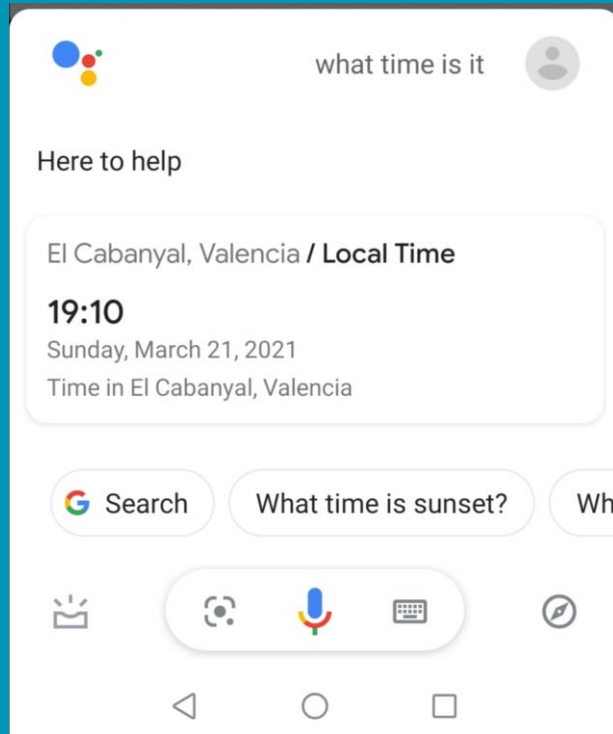
Recurrent NN



Estado Oculto



## RNN



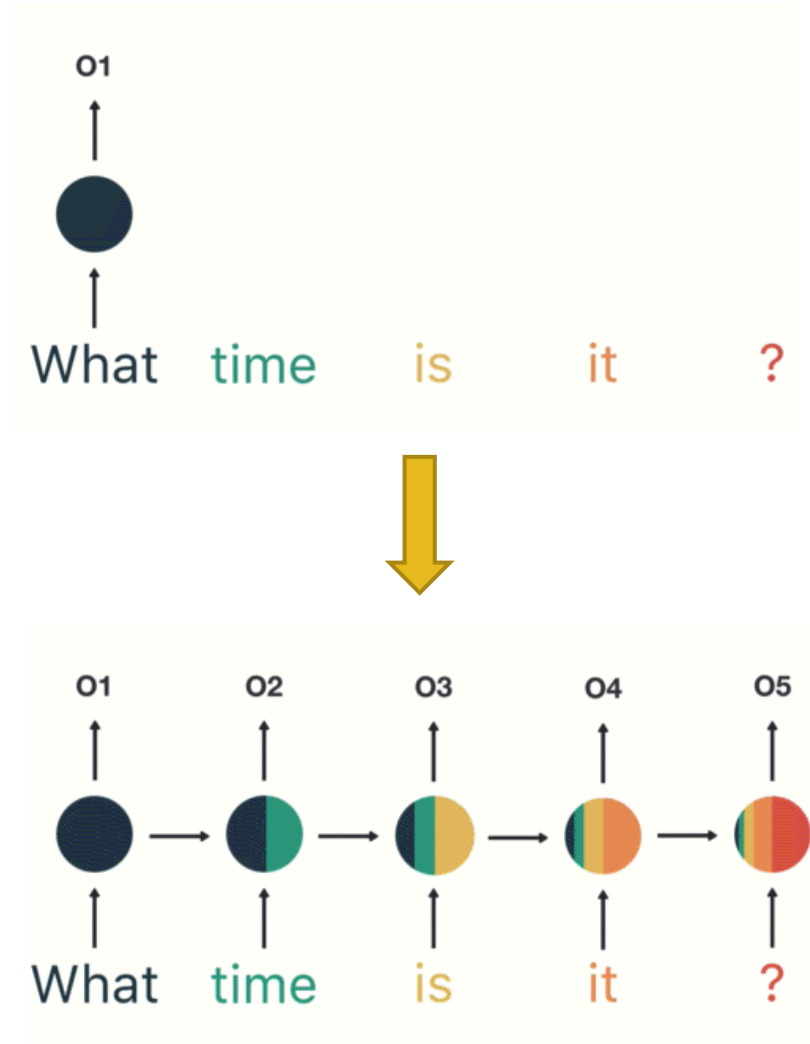
1) Dividimos la frase en Palabras individuales.

What time is it ?

## RNN

2) Alimentamos la RNN con la palabra “time” y el estado oculto del paso anterior.

3) Repetimos el proceso hasta el final. Pasamos el último estado a una capa FF para clasificar la intención



## RNN - Problema de Memoria

- Debido al desvanecimiento del Gradiente Descendiente, esta red neuronal tiene problemas de “memoria a corto plazo”.
- A medida que procesa más pasos, le cuesta más retener información de los pasos iniciales.
- La RNN NO aprende bien las dependencias de largo alcance a través de los pasos del tiempo



Estado Oculto final de la RNN

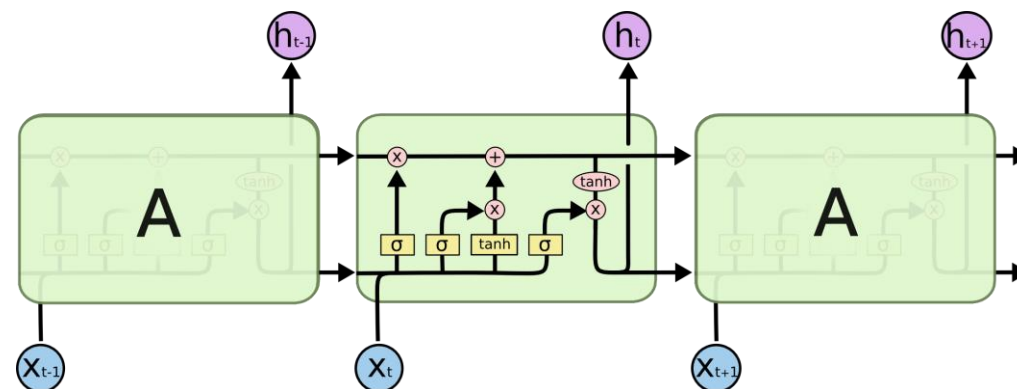
## LSTM

Las RNN sufren de memoria a corto plazo. ¿Cómo se soluciona?

LSTM funciona como una RNN, pero es capaz de aprender dependencias a 'largo plazo' utilizando mecanismos llamados '*puertas*' (gates).

Estas puertas son capaces de aprender qué información pueden añadir o eliminar de los estados ocultos.

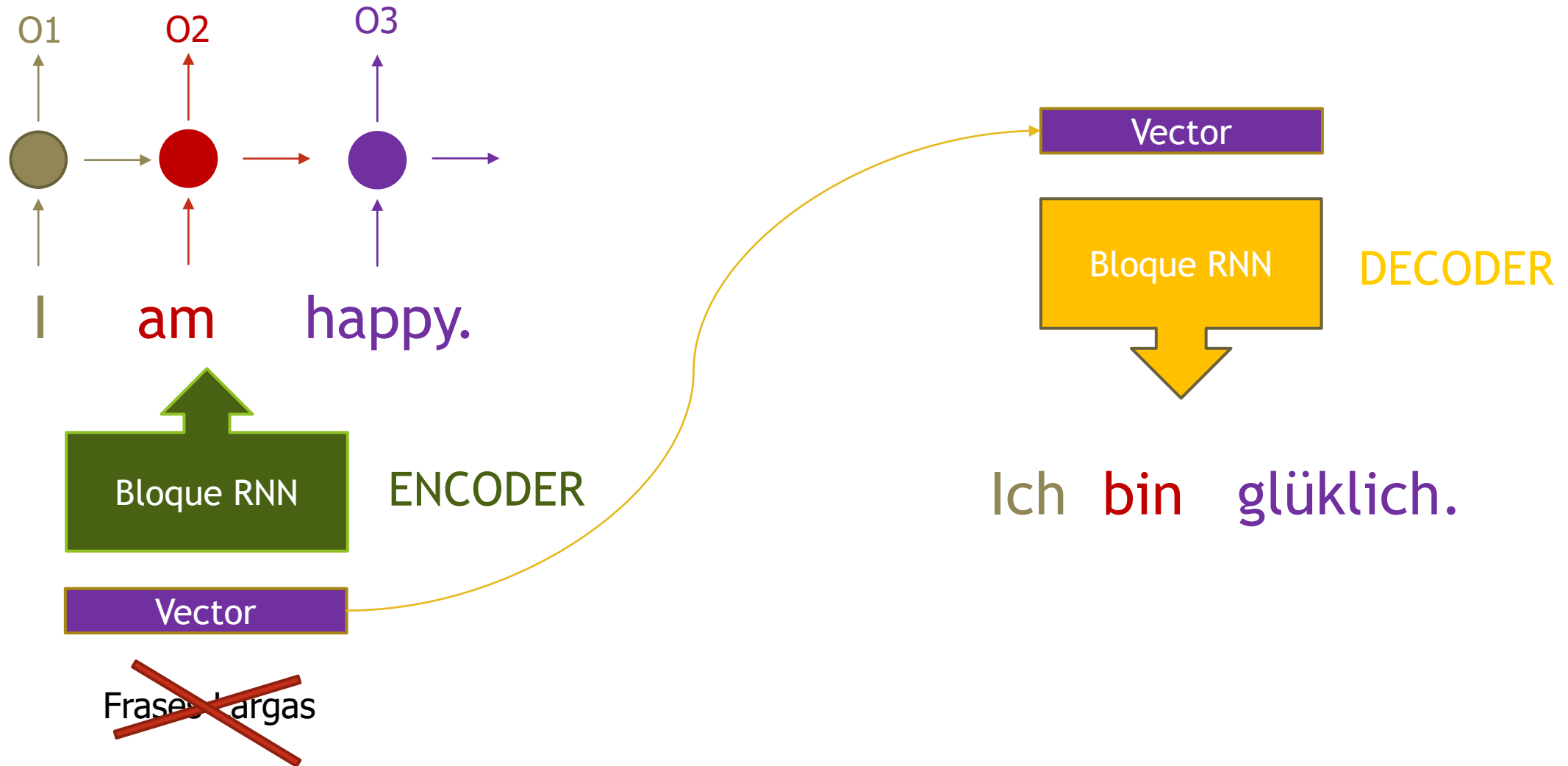
<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>



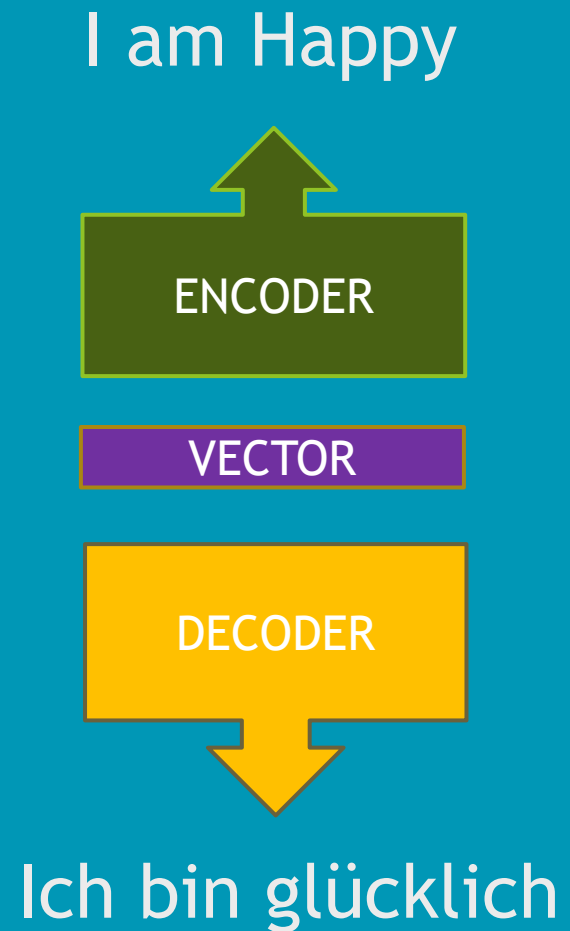
## Transformers



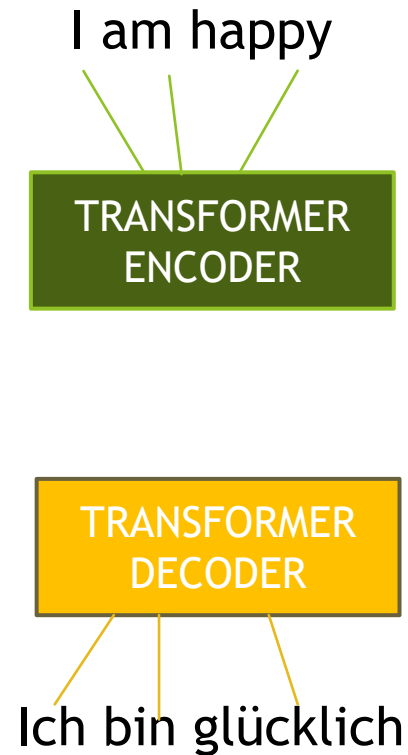
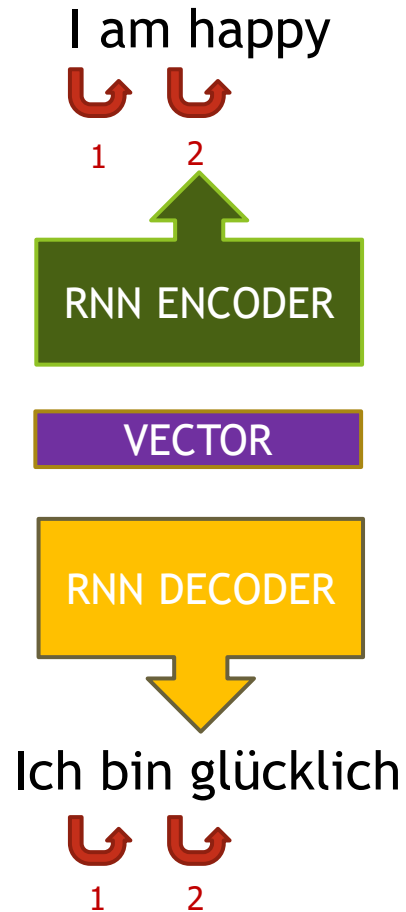
# NLP



# Encoder - Decoder



## ¿Qué lo diferencia de los Transformers?



1) El Encoding y el Decoding se realiza de forma secuencial en las LSTM

2) En La arquitectura de Transformers, el proceso se realiza en paralelo.



## ¿Cómo Funciona?

- La capa de Encoder es un *Stack* de encoders en capas
- Permite ejecutar varias veces la secuencia
- Podemos capturar las propiedades de la frase primero

I am Happy

TRANSFORMER ENCODER

TRANSFORMER ENCODER

TRANSFORMER ENCODER

TRANSFORMER ENCODER

POS Tags  
Constiuyentes  
Dependencias  
Roles semánticos  
Coreferencias, etc.,

TRANSFORMER DECODER

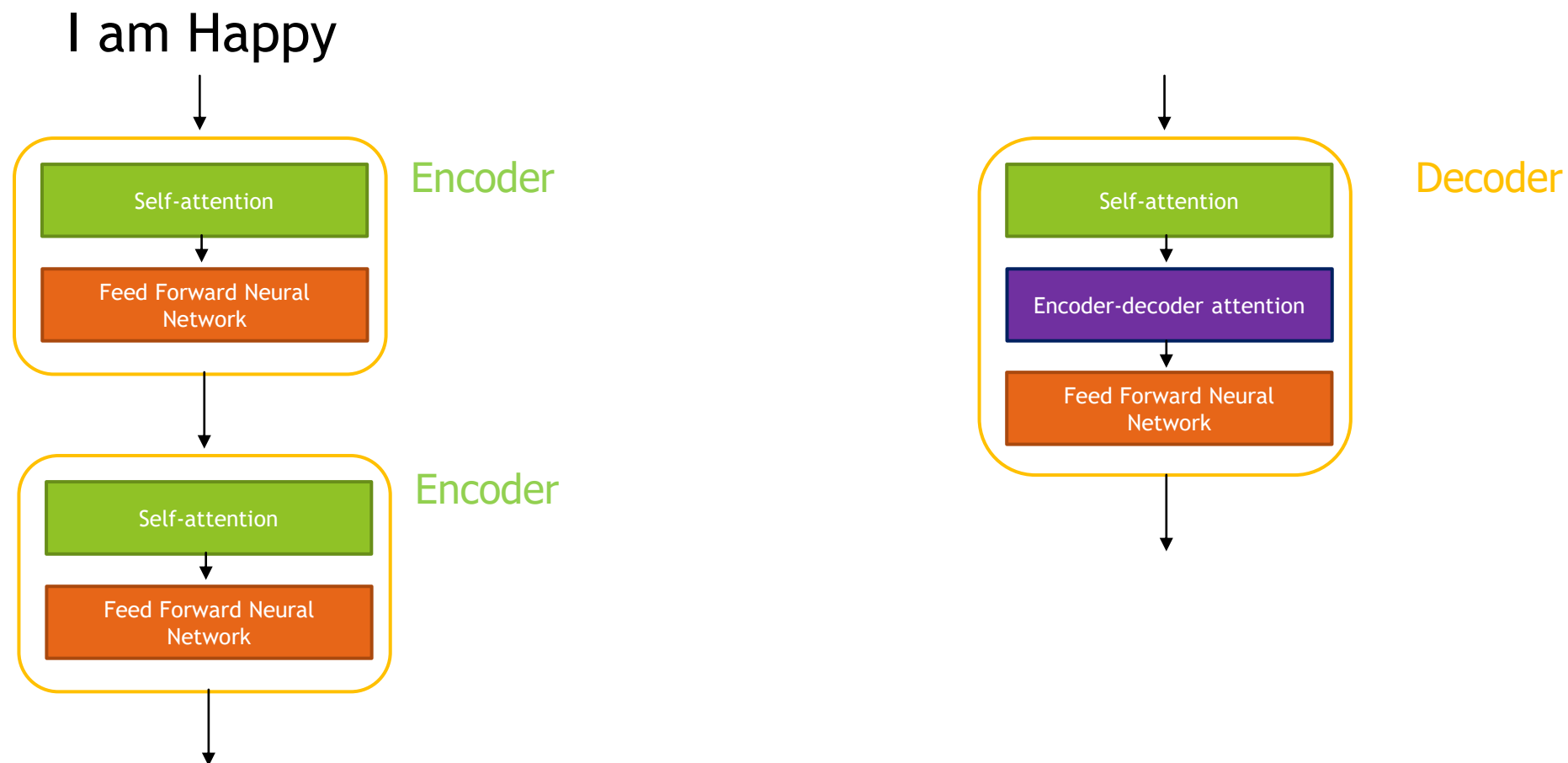
TRANSFORMER DECODER

TRANSFORMER DECODER

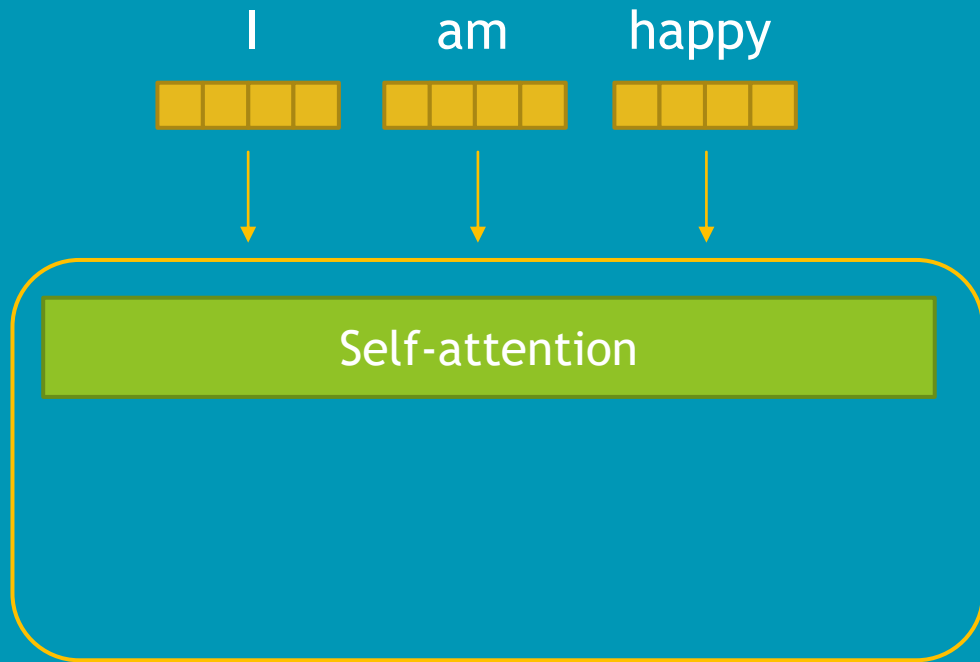
TRANSFORMER DECODER

Ich bin glücklich

## Transformer Encoder/Decoder



## Transformer Encoder-Decoder



### Attention

The Rabbit ran because I scared it  
Der Hase rannte, weil ich ihn erschreckt hatte

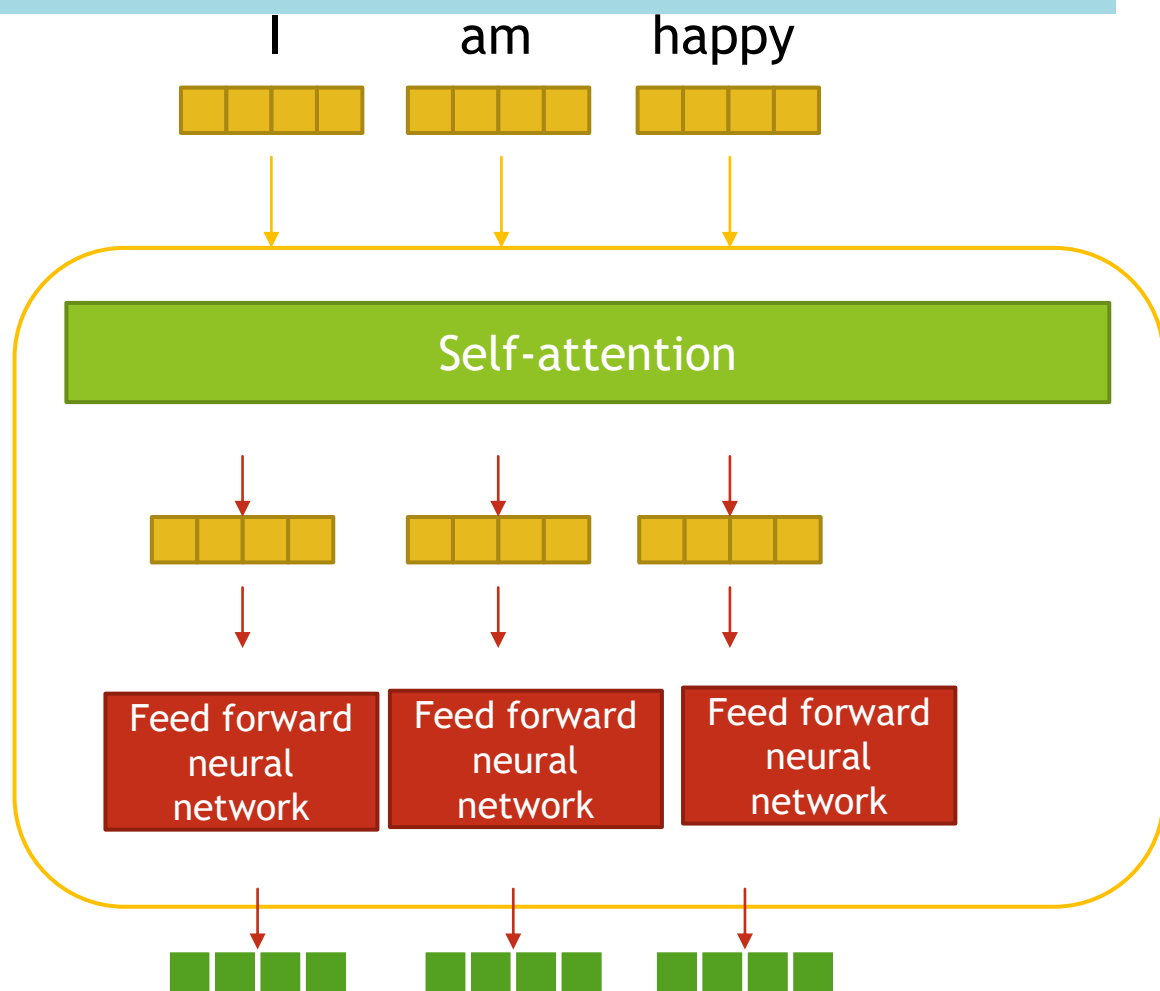
This diagram illustrates cross-attention. It shows two sentences: "The Rabbit ran because I scared it" in English and "Der Hase rannte, weil ich ihn erschreckt hatte" in German. Purple arrows indicate the attention mechanism: one arrow points from the word "I" in the English sentence to the word "ich" in the German sentence, and another arrow points from the word "scared" in the English sentence to the word "erschreckt" in the German sentence.

### Self-attention

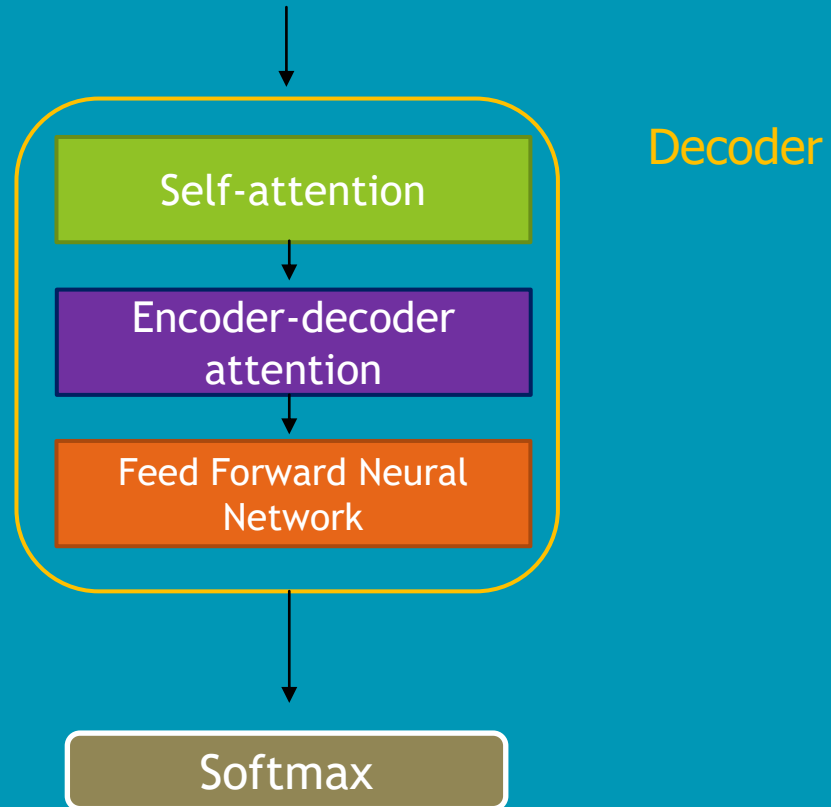
The Rabbit ran because I scared it  
The Rabbit ran because I scared it

This diagram illustrates self-attention. It shows the same English sentence "The Rabbit ran because I scared it" twice. Yellow arrows indicate the attention mechanism within the first instance: one arrow points from the word "scared" to the word "it", and another arrow points from the word "scared" to the word "Rabbit".

# NLP



- La capa self-attention analiza las dependencias entre palabras
- Luego pasa los vectores a las Feed Forward NN
- Las Feed Forward Neural Networks envían el output al siguiente encoder
- La PARALELIZACIÓN permite acelerar el proceso de entrenamiento.
- Los transformers trabajan relativamente bien en grandes cantidades de Texto.



Ich: 0.9  
du: 0.05  
hase: 0.004  
...



Ich: 0.0001  
du: 0.015  
bin: 0.85  
...



Ich: 0.002  
du: 0.005  
bin: 0.0445  
Glück: 0.92  
...



Ich: 0.00026  
du: 0.005  
bin: 0.0003  
<STOP>: 0.99  
...

Ich bin glücklich <STOP>

# Generative Pre-trained Transformer 3. GPT-3

- Desarrollado por Open-AI. Mayo 2020.
- Modelo Generador de Texto. Modelo predicción del Lenguaje.
- Basados en la arquitectura de los Transformer.
- Muy polémico desde sus inicios. No es público.
- Es una evolución del GPT-2, publicado en 2019.

William Shakespeare ¶

**Transformer AI poetry: Poetry classics as reimaged and rewritten by an artificial intelligence.**

**"Sonnet 18"**

**William Shakespeare**

**Shall I compare thee to a summer's day?**

**Thou art more lovely and more temperate:**

**Rough winds do shake the darling buds of May,**

**And summer's lease hath all too short a date;**

A winter's day, when beams of sun are few,

And skies are grey with clouds intemperate,

Is to thy beauty somehow sordid grown.

Her lovely-lovely eye is quite shut up,

Like windows pale before a sunny shower.

The earth that's cold and moist, like to my mistress,

When once that she unbrac'd, her lovely jewel,

Grows as both ugly and uncourtly.

Poor soul, the center of my motley host,

See, now they request thy sovereign presence;

The din of merry hearts hath brought me thus

To greet thee, thou unsoil'd and saint-like beauty.

A Translation by an Artificial Neural Network that draws inspiration from François Connor's translation.

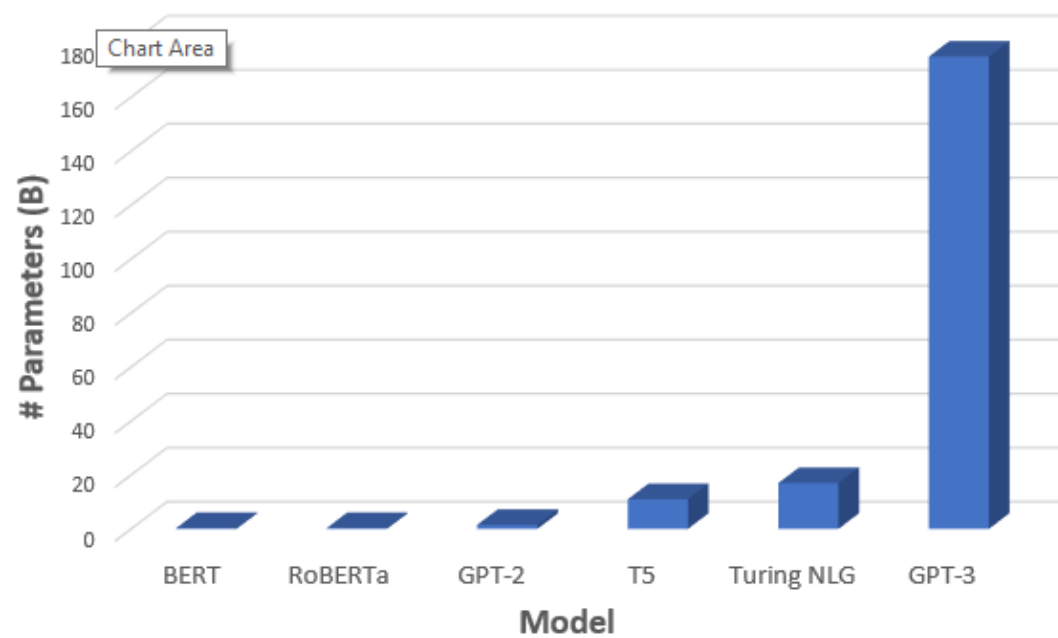
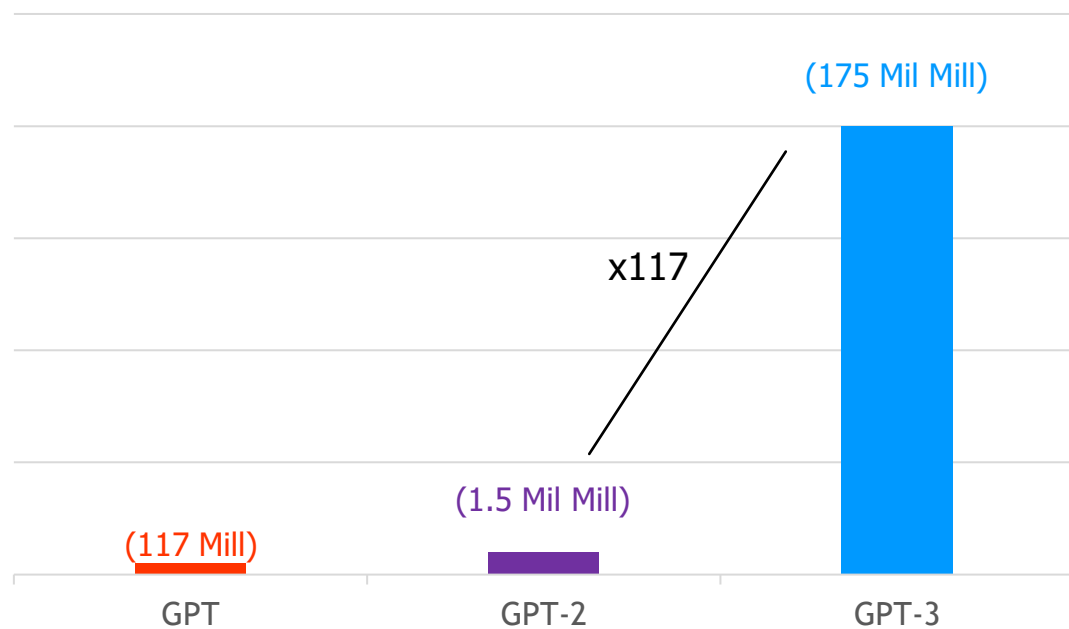
Soften thou, for showers often stain the daylight,

And longer they torment my Mary's day,

# NLP

## GPT-3

Nº de Parámetros



## GPT-3 Ejemplos

- 1) ML Generador de Código en Keras
- 2) Diseño de páginas web con una URL y una descripción.
- 3) Reescritura de estilo y finalización de textos
- 4) Convertir de Inglés a comandos de Linux
- 5) Lenguaje Natural a SQL



FIN