

# NLP: Preprocesamiento de Lenguaje Natural

## Clasificación de Texto

### Definición

Problema de clasificación convencional donde la entrada es texto en lenguaje natural. Sirve para categorizar de forma automática el extracto de un texto.

### Implementación

- **Reglas** → escritas a mano, construir y mantenerlas puede ser costoso
  - entradas:
    - un documento
    - un conjunto prefijado de clases
  - salidas:
    - una clase perteneciente al conjunto
- **Aprendizaje automático supervisado**
  - entradas:
    - un documento
    - un conjunto prefijado de clases
    - un conjunto de documentos clasificados
  - salidas:
    - un clasificador entrenado
  - Tipos de Clasificadores
    - Naïve Bayes
    - Logistic Regression
    - Support-Vector Machines
    - K-Nearest Neighbors

### Aprendizaje Bayesiano

Un documento pertenece a aquella clase que maximice su probabilidad condicional.

Problema → Ingenuo

Asume que el orden de las palabras no importa y son independientes entre sí.

### Bag Of Words

Convierte los textos en números.

### Armado de Vocabulario

1. Filtrar palabras (si usamos sólo algunas en lugar de todas → Regex)
2. Contamos todas las palabras que aparecen en los documentos
3. Armamos una lista de palabras únicas

## Tokenización

Partir un texto utilizando caracteres específicos → obtenemos palabras o aproximaciones a palabras

## Laplace Smoothing

Cuando aparece una palabra nueva.

## Fórmulas

$$P(\text{"palabra"} | \text{clase}) = \frac{\#\text{"palabra"} \text{ en documentos de clase "clase"} + 1}{\#\text{palabras en documento de clase "clase"} + \#\text{palabras del vocabulario}}$$

$$P(\text{clase} | \text{test}) = P(\text{clase}) * \prod_{i=1}^n P(\text{"palabra}_i \text{ en test} | \text{clase}), n: \#\text{palabras en test}$$

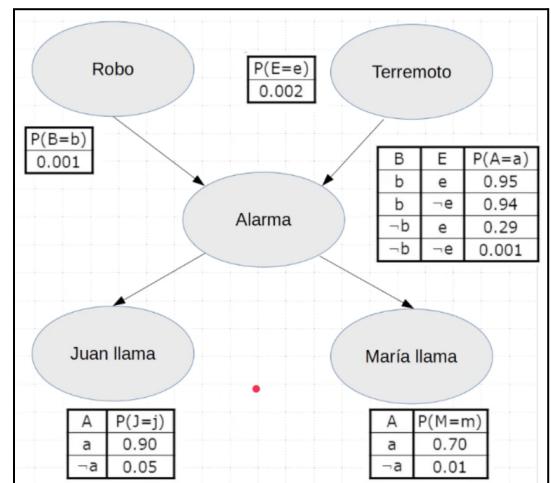
$P(\text{clase A} | \text{test}) > P(\text{clase B} | \text{test}) \Rightarrow \text{por Bayes el documento de test habla de A}$

## Redes Bayesianas

- Grafo acíclico dirigido
- Los nodos representan variables
- Las aristas representan dependencias condicionales
- Tienen asociada una tabla con probabilidades condicionales por cada nodo
- Permiten realizar inferencia según la observación de un evento

## vs Bayes Naïve

- las redes bayesianas
  - permiten inferencias mucho más precisas
  - son más complejas de construir y mantener
- bayes naïve
  - es muy rápido
  - requiere poco almacenamiento
  - robusto ante características irrelevantes
  - bueno donde hay muchas características y todas son relevantes
  - si las palabras son independientes, es óptimo



# Análisis de Sentimientos

## Definición

Tarea de clasificación de textos. Detección de actitudes. Extracción de opiniones en base a aspectos o atributos.

## Tareas

- Sencillas: Determinar polaridad de la actitud (positiva, negativa).
- Intermedias: Puntar la actitud de 1 a 5 (estimar la confianza del consumidor, predecir el mercado de valores).
- Complejas: Detectar portador, destinatario y tipo de actitud (amor, odio, deseo, valoración).

## Tipología de Scherer de los estados afectivos

- ★ **Emoción** → ira, tristeza, alegría, miedo, vergüenza, orgullo, alegría
- ★ **Estado de ánimo** → alegre, triste, irritable, apático
- ★ **Postura interpersonal** → distante, frío, cálido, de apoyo, de desprecio
- ★ **Actitudes** → simpatía, amor, odio, deseo, valoración
- ★ **Rasgos de personalidad** → nervioso, ansioso, imprudente, taciturno, hostil

## Algoritmo de Pang & Lee: Detección de polaridad

### Pasos

1. Tokenización del texto → partimos el texto en palabras
2. Extracción de características → palabras o frases claves
3. Clasificación
  - a. Naïve Bayes (Multinomial Binarizada)
    - eliminar las palabras duplicadas (mejores resultados)
    - problemas: sutilezas y expectativas frustradas
  - b. MaxEnt
  - c. SVM
  - d. Logistic Regression
  - e. KNN
4. Entrenamiento → CV para encontrar el conjunto que optimiza los valores

## Lexicón de Sentimientos

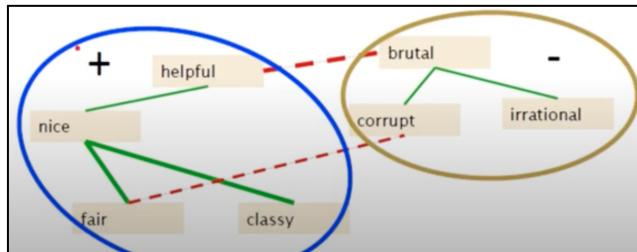
Lo entrenamos si no queremos entrenar un algoritmo de clasificación. Diccionario que brinda la carga de valor de una palabra.

## Creación de un Lexicón

- un conjunto de ejemplos previamente clasificados
- reglas que identifiquen patrones en una frase

## Algoritmo de Hatzivassilogloy & McKeown para la ampliación de un Lexicón

- Adjetivos unidos por "y" tienen la misma polaridad
- Adjetivos unidos por "pero" tienen opuesta polaridad



- Necesita un paso extra de revisión de datos obtenidos

## Algoritmo de Turney para obtener la polaridad de las frases

### Pasos

#### 1. Extraer frases y armar un Lexicón de frases

Primer Palabra	Segunda Palabra	Tercer Palabra (no se extrajo)
Adjetivo	Sustantivo (plural o singular)	Cualquier palabra
Adverbio	Adjetivo	No Sustantivo
Adjetivo	Adjetivo	No Sustantivo
Sustantivo (plural o singular)	Adjetivo	No Sustantivo
Adverbio	Verbos	Cualquier palabra

#### 2. Aprender la polaridad de cada frase → qué tan probable es que dos palabras estén juntas o separadas

$$PMI(p1, p2) = \log_2 \left( \frac{\#(p1 \text{ near } p2)}{\#p1 * \#p2} \right)$$

$$\text{Polaridad(frase)} = PMI(\text{frase}, p1) - PMI(\text{frase}, p2) = \log_2 \left( \frac{\#(\text{frase near } p1) * \#p2}{\#(\text{frase near } p2) * \#p1} \right)$$

#### 3. Puntuar el texto según el promedio de la polaridad de cada frase

$$\frac{\sum_{i=1}^n \text{Polaridad}(\text{frase}_i)}{n}, \text{ } n: \text{cantidad total de frases}$$

## Aspectos

### Método de Minqing Hu y Bing Liu

- ★ Frecuencias → las frases frecuentes se denominan aspectos, atributos u objetos de sentimiento (representan al objeto que se está criticando)
- ★ Reglas → se aplican sobre las frases frecuentes para detectar sentimientos
- El aspecto puede no ser nombrado
- Es posible utilizar una clasificación supervisada
- Hay que entrenar un clasificador
- Si la cantidad de críticas no está balanceada, no se puede utilizar el estimador de precisión y hay que utilizar F-Score
- Si el desbalanceo es muy pronunciado se puede degradar el rendimiento del clasificador
- Hay aspectos de grano grueso y aspectos de grano fino

# Extracción de Información

Capturar información fáctica y transformarla en estructurada.

## NER: Reconocimiento de Nombres de Entidades

Cualquier cosa susceptible de tener un nombre propio.

### Etiquetamiento Secuencial

1. Obtener un conjunto de datos representativos al dominio
2. Etiquetar cada palabra con la clase que le corresponde
  - ❖ IO-Encoding → asigna a cada palabra una categoría
  - ❖ IOB-Encoding → pone una B si es la primera palabra o I si es  $\geq$  la segunda

Tokens	IO - encoding	IOB - encoding
Fred	PER	B-PER
Showed	O	O
Sue	PER	B-PER
Mengqiu	PER	B-PER
Huang	PER	I-PER
's	O	O
new	O	O
painting	O	O

3. Especificación de características a utilizar

- Basadas en Palabras
  - palabra actual (token)
  - palabra previa o siguiente
  - substring de una palabra
  - forma de una palabra (nomenclatura)
- Basadas en otro tipo de inferencia lingüística
  - etiquetado gramatical (adj + sust + sust)
- Contexto de etiquetado
  - etiquetado anterior y siguiente (persona + persona + ...)
- Algoritmos de Inferencia
  - Greedy Inference
  - Beam Inference
  - Viterbi Inference
  - CRFS

4. Entrenamiento de un clasificador secuencial para predecir las etiquetas

## Extracción de Relaciones Semánticas

### Ontología

- ❖ “Is-a” (hipónimo)
- ❖ “Instance of” (Buenos Aires es una instancia de una ciudad)
- Forma de representar una estructura de conocimiento

## Construcción

Reglas escritas a mano (pattern-matching)

- + muy preciso
- + adaptable a dominios específicos
- muy bajo recall
- mucho trabajo manual
- se puede mejorar con otros métodos

Aprendizaje automático supervisado

1. decidir qué relaciones son interesantes
2. detectar entidades
3. encontrar un conjunto de datos
  - a. etiquetar entidades
  - b. etiquetar relaciones entre entidades
  - c. separar train-test
    - para evaluar desempeño: precision, recall, F1
4. entrenar un clasificador

Aprendizaje Supervisado (etiquetamiento manual)

- Bag of Words: palabras con frecuencia muy alta sin valor
- Clasificadores:
  - ◆ Bayes Naïve
  - ◆ MaxEnt
  - ◆ SVM

Auto-Supervisados

Generan su propio conjunto de entrenamiento etiquetado

- ★ Bootstrap
  - tener un par de entidades semilla cuya relación es conocida
  - extraer el contexto de la oración encontrada y reemplazarlas por comodines
  - realizar búsquedas con esos patrones para encontrar nuevas entidades
  - repetir
- ★ Algoritmo de Dripe
  - mejora de bootstrap
  - toma la parte común de los patrones
- ★ Distant Supervision
  - combina bootstrapping con aprendizaje supervisado
  - con un gran conjunto de datos busca patrones y extrae características
  - En común con los Supervisados
    - usa un clasificador con varias características
    - no itera N veces para extraer patrones
  - En común con los No Supervisados
    - usa grandes cantidades de datos sin etiquetar
    - no es sensible a cómo se generó el corpus
  - Pasos

1. genera una tupla para cada relación
2. detecta las sentencias donde están las tuplas
3. extrae características frecuentes
4. entrena un clasificador utilizando los patrones

No Supervisados para la Web

★ Know it all

- independiente del dominio
- escalable

★ Text Runner

- detecta todas las relaciones que cree que existen
- 1 sola pasada
- extrae relaciones semánticas no definidas

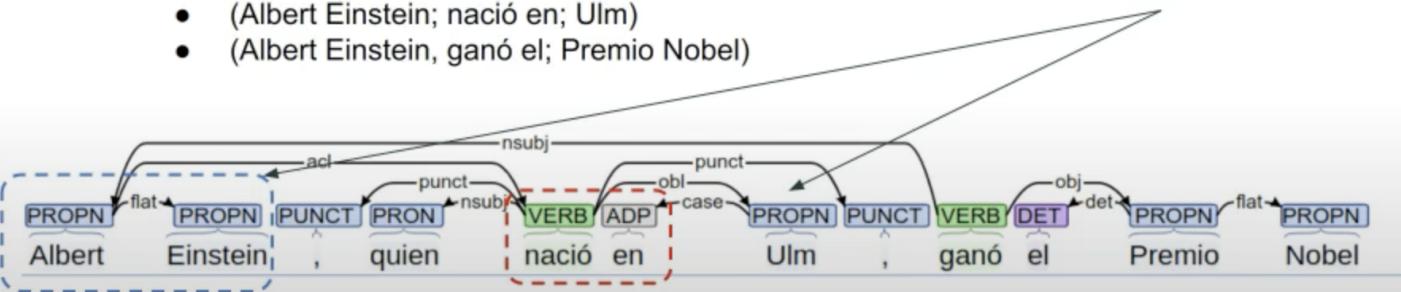
★ ECMes

- análisis superficial
- reconocimiento de nombres de entidades
- árbol dependencias sintácticas
  - ❖ utiliza un conjunto de datos etiquetados con un formato específico
  - ❖ crea patrones y los busca

Albert Einstein, quien nació en Ulm, ganó el Premio Nobel

Crea patrones, para buscar en el árbol

- (Albert Einstein; nació en; Ulm)
- (Albert Einstein, ganó el; Premio Nobel)



# Redes de Aprendizaje Profundo

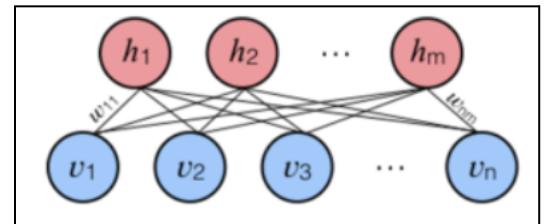
- *profundas* → tienen muchas capas
- el entrenamiento es más complejo → resuelven problemas más complejos
- cada capa resuelve un problema reducido
- son muy costosas en tiempo de entrenamiento

## Funcionalidades

- Aprendizaje no supervisado → extracción de patrones
- Aprendizaje supervisado → patrones etiquetados
  - ◆ procesamiento de texto
  - ◆ reconocimiento de imágenes
  - ◆ reconocimiento de texto
  - ◆ reconocimiento del habla
- Clasificación
- Análisis de series de tiempo

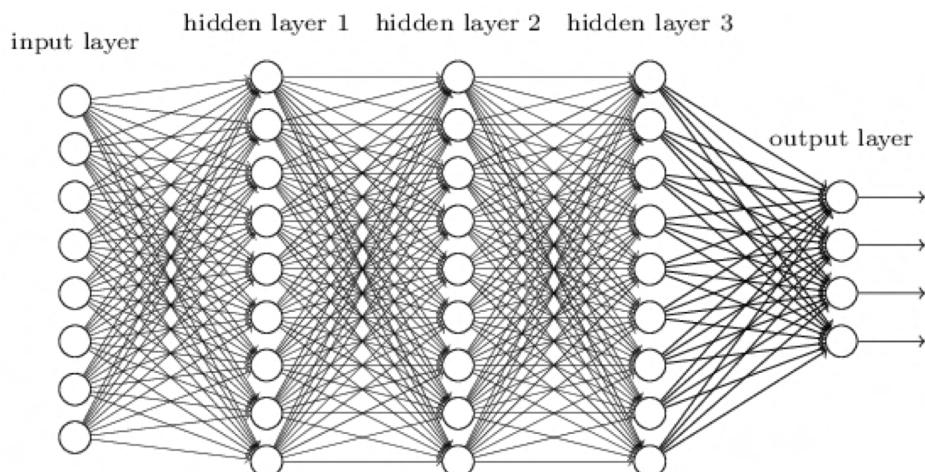
## RBM: Restricted Boltzmann Machines

1. Ejecutar una entrada en sentido directo
2. Ejecutar en sentido inverso
3. Comparar con KL Divergence y ajustar pesos y bias hasta que las salidas de la red invertida coincidan con las entradas



## DBN: Deep Belief Nets

Construye una RBM y apila la red



- Se entrena cada 2 capas
- Cada capa aprende el input entero
- La red aprende a detectar patrones inherentes en los datos
- Segunda parte de entrenamiento supervisado con cantidad pequeña de datos

# Autoencoders

- ❖ RBM
- ❖ Deep Belief Nets

## Definición

Mecanismo que aprende a producir a la salida exactamente la misma información que recibe a la entrada. La entrada y la salida tienen el mismo número de neuronas (compresor, reducción de la dimensionalidad, eliminación de ruido).

## Entrenamiento

Con Backpropagation mediante la métrica *loss* → cantidad de información que la red perdió al reconstruir el input.

# CNN: Convolutional Neural Nets

- Dominan la visión espacial
- Replican el funcionamiento del cerebro humano

## Capa Convolucional

Una neurona conectada con todas las de entrada (entre ellas no están conectadas).

## Capa Pooling

Hace una reducción de la dimensionalidad.

- ★ Max Pooling
- ★ Average Pooling

## Capa FC (Fully Connected)

Perceptrón de 2 capas que clasifica la salida de la red. No está conectada a las neuronas de entrada.

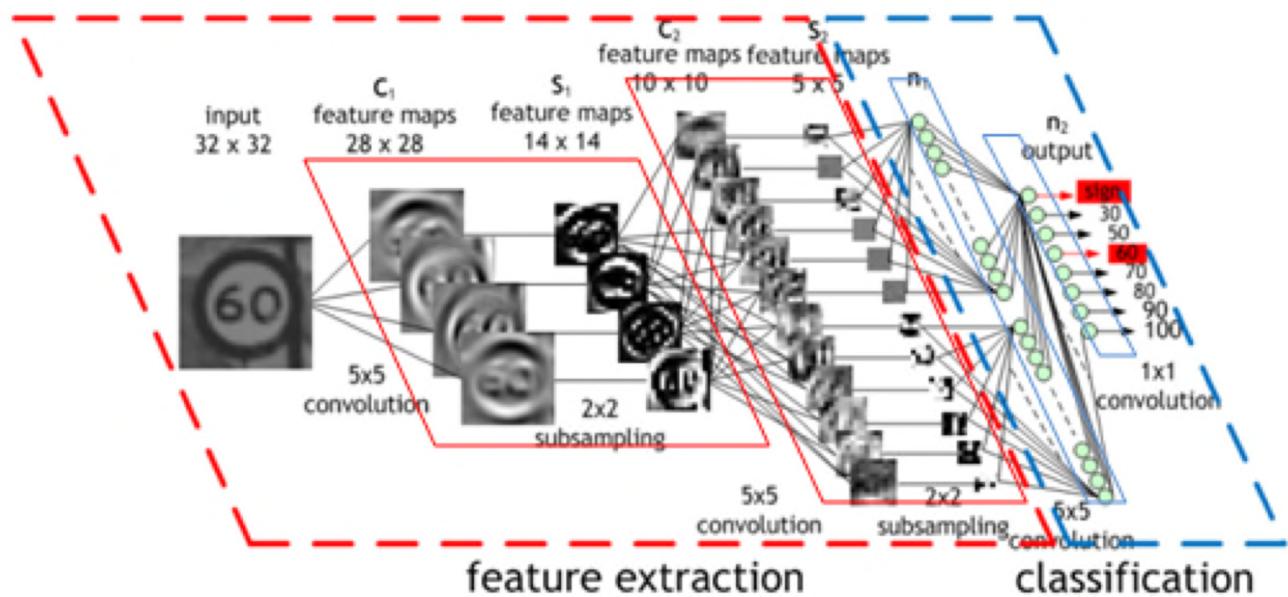
## Arquitectura

Múltiples capas convolucionales, RELU y Pooling pero una sola FC.

## Convolución

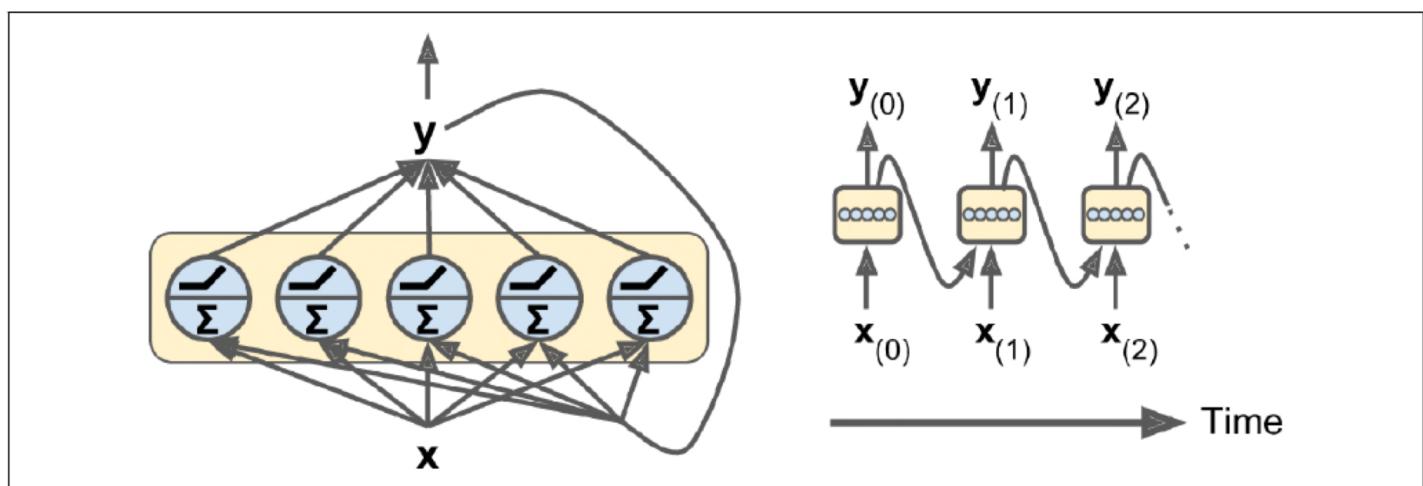
Manera de combinar 2 funciones en 1 nueva.

## Ilustración



## RR: Redes Recurrentes

- La salida vuelve a entrar → bucles de retroalimentación
- Pueden recibir una secuencia de valores como entrada y devolver una secuencia de valores de salida
- Pueden tener no solo una neurona sino una capa entera
- Útiles cuando los patrones en los datos cambian en el tiempo



## Celdas de Memoria

La salida actual depende de la anterior. Así se llama a la neurona de la red.

## Recurrentes vs Hacia Adelante

- Recurrentes → predicción
- Hacia Adelante → clasificación o regresión

## Tipos

Secuencia → Secuencia (Series temporales)

- Clasificación de videos cuadro a cuadro
- Precio de acciones en el futuro

Secuencia → Singular

- Clasificación de documentos
- Análisis de Sentimientos

Singular → Secuencia

- Etiquetado de imágenes

## Entrenamiento

Backpropagation through time → desplegarla en el tiempo y usar backpropagation

## Métricas

- ★ Naïve → error cuadrático medio entre el último valor predicho y el real

## Implementación con Keras

1. Crea un modelo secuencial
2. Crea la capa recurrente

## Problemas RNR y RNR Profundas

- **Desvanecimiento del gradiente**
  - como todas las redes profundas (cuanto más capas, el gradiente tiende a 0 y la actualización de los pesos es cada vez peor)
  - el problema es exponencial pues hay varios pasos temporales
- **Gradientes inestables** → el gradiente no converge, oscila, cambia la función de error en cada paso y se dificulta que la red converja
- **Demoras** → una sola capa con 100 pasos temporales es como entrenar una red de 100 capas de alimentación hacia adelante
- **Memoria a corto plazo** → no pueden recordarse secuencias demasiado largas

Soluciones: Celdas LSTM, GRU, GRU + Redes Convolucionales

## LSTM: Long Short-Term Memory

Como una celda tradicional pero

- + se desempeña mejor
- + el entrenamiento es más rápido
- + puede detectar cadenas mucho más largas
- + le indica a la red cuándo olvidar y recordar el input

## GRU: Gated Recurrent Unit

Versión simplificada de la celda LSTM

## GRU + Convolucionales

Para mejorar la memoria de las celdas. Las capas convolucionales son 1D y se utilizan kernel de 1D.

## Generación de Texto



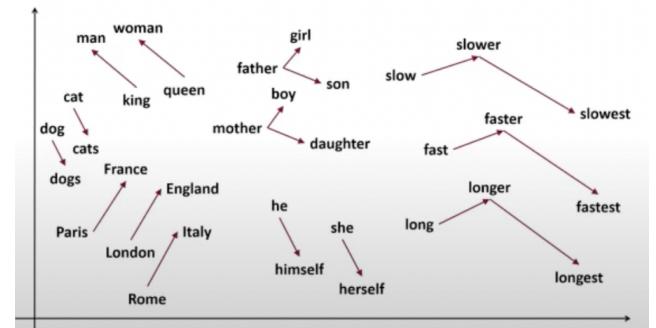
### Construcción del modelo:

```
model = keras.models.Sequential([
    keras.layers.GRU(128, return_sequences=True, input_shape=[None, max_id], dropout=0.2, recurrent_dropout=0.2),
    keras.layers.GRU(128, return_sequences=True, dropout=0.2, recurrent_dropout=0.2),
    keras.layers.TimeDistributed(keras.layers.Dense(max_id, activation="softmax"))
])
model.compile(loss="sparse_categorical_crossentropy", optimizer="adam")
history = model.fit(dataset, epochs=20)
```

128 son las neuronas de salida de las capas GRU  
La capa de salida es **Densa** distribuida en el tiempo y tiene 39 neuronas de salida, activadas con **Softmax**. Cada neurona corresponde a un único carácter, pero cómo para un texto de entrada podría haber más de un carácter posible utilizamos **Softmax**.

## Embeddings

- Reducción de la dimensionalidad → paso vectores de one hot encoding por una red perceptrón y genero un vector de salida denso
- Conserva propiedades de las palabras



## Redes de Tensores Recursivas

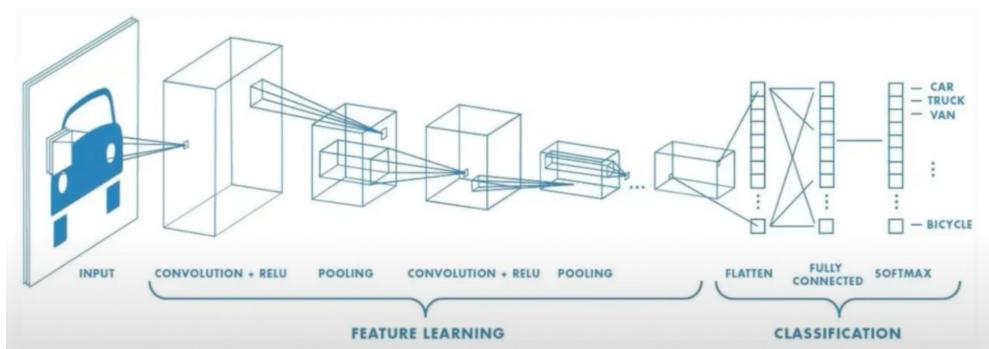
- Tareas de *Análisis de Sentimiento* y *Parseo de Imágenes* → tiene en cuenta el orden y la agrupación sintáctica
- Estructura de árbol → la raíz genera un score que es ingresado recursivamente en las hojas
- Funciona para *Etiquetado Gramatical*
  - frase nominal
  - frase verbal
- Utiliza backpropagation → árbol obtenido vs árbol generado manual

## Redes Deep Fakes

Funcionan con [RBM](#), [Autoencoders](#) y [DBN](#): Reconocimiento de Caras.

## GAN: Generative Adversarial Networks

- Redes Convolucionales → Capa de Activación: SoftMax



## SoftMax

- Pensada para múltiples salidas
- La clasificación puede ser múltiple (se pueden activar varias neuronas a la vez)

## Características

- ❖ Redes no supervisadas
- ❖ No arman clusters
- ❖ No clasifican datos
- ❖ No resuelven problemas de regresión
- ❖ Generan datos nuevos

## Usos

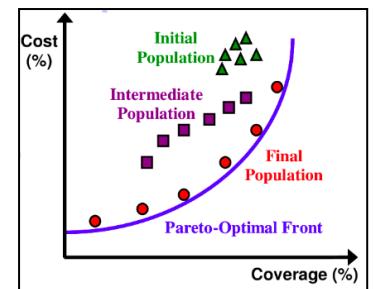
- Reconstrucción y transformación de imágenes

## ProGAN

- genera imágenes de alta calidad
- su capacidad para controlar características específicas de la imagen generada es limitada
- modificar levemente la entrada afecta varias funciones al mismo tiempo (modificación genética)

# Algoritmos Genéticos

- Fenotipo: Problema a solucionar
- Cromosoma: Cada una de las soluciones posibles
- Gen: Cada uno de los símbolos en el cromosoma
- Genotipo: Si los símbolos son binarios
- Generación: Cada iteración
- Tamaño de la Población: Cantidad de soluciones
  - Excesiva → lento
  - Escasa → solución poco óptima



## Selección

### Por Ruleta

- Método azaroso
- Cromosoma
  - probabilidad según utilidad
  - porción de ruleta
- La ruleta gira y donde caiga es el método elegido
- Las mejores soluciones tienen más probabilidades de salir elegidas

### Elitista

Copiamos a los mejores en la siguiente generación, el resto por ruleta.

### Jerárquica

Relación jerárquica entre soluciones.

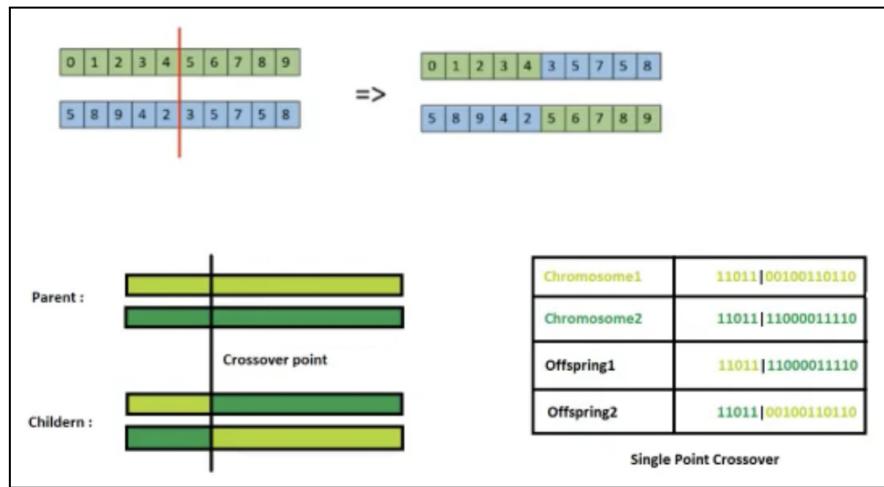
### Por Torneo

Competen las soluciones.

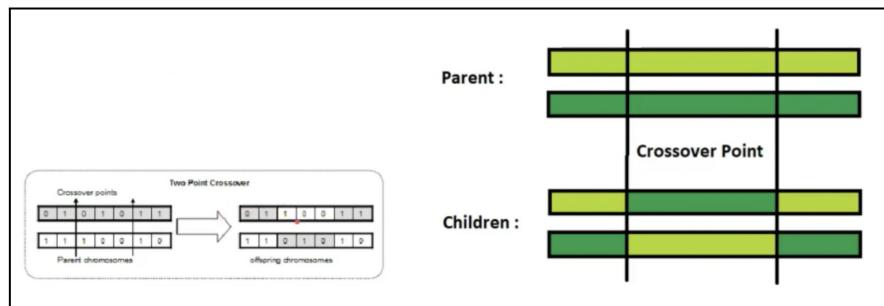
### Cruza

#### Crossover 1 Punto

- 2 soluciones (cromosomas) → 1 punto al azar → corto
- pego una parte de la solución en la otra y viceversa

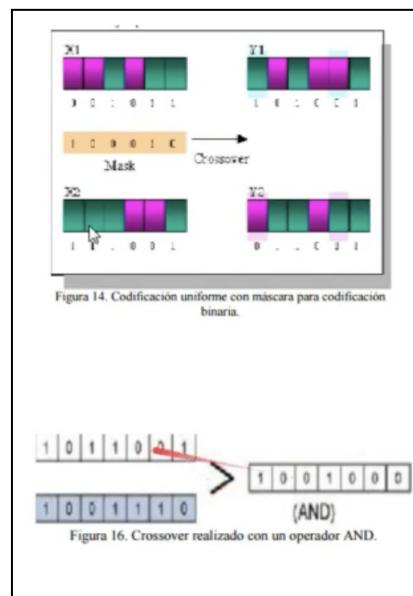


## Crossover 2 Puntos



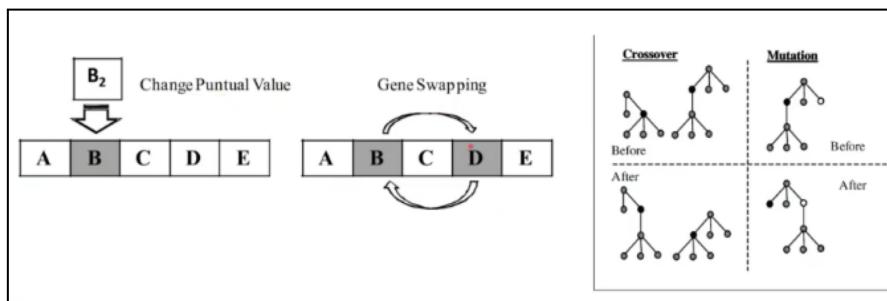
## Crossover Aritmético

- Se pueden cortar en distintas partes los padres



## Mutación

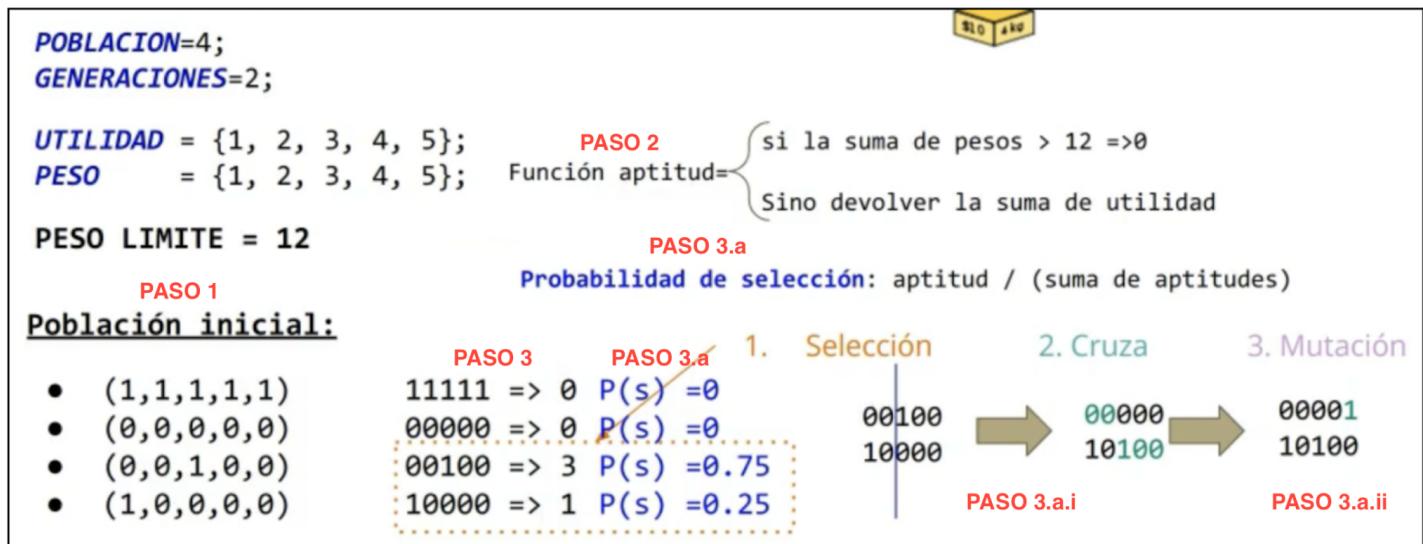
No se debe abusar → no tienen que ser muy poquitas las mutaciones si no la solución no converge. Consiste en cambiar aleatoriamente el gen.



## Implementación

1. Comenzar con una población  $P$  generada aleatoriamente de  $n$  cromosomas de  $k$  bits
  2. Calcular la aptitud  $f(x)$  para cada cromosoma  $x$  de  $P$
  3. Repetir hasta que se hayan creado  $n$  descendientes
    - a. Seleccionar un par de cromosomas de  $P$ , siendo la probabilidad de selección una función creciente de la aptitud
      - i. Con probabilidad  $pc$  (probabilidad de cruce), cruzar el par en un punto elegido aleatoriamente para formar dos descendientes y agregarlos a una nueva población  $P'$
      - ii. Mutar los dos descendientes en cada lugar con probabilidad  $pm$  (probabilidad de mutación) y colocar los cromosomas resultantes en la nueva población  $P'$
      - iii. Si  $n$  es impar, se puede rechazar aleatoriamente a un miembro de la nueva población
  4. Reemplazar la población actual  $P$  con la nueva  $P'$
  5. Volver al paso 2

## Ejemplo



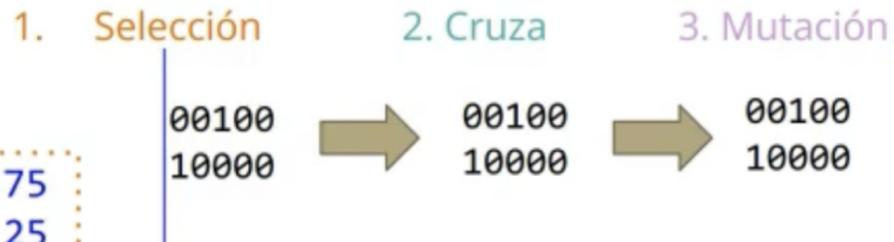
### Población Iteración 0:

4, 5};  
4, 5};

**Repetimos, hasta obtener 4 elementos en la población**

- (0,0,0,0,1) de la mutación
- (1,0,1,0,0)
- (0,0,1,0,0)
- (1,0,0,0,0) de la población inicial

11111 => 0  $P(s) = 0$   
 00000 => 0  $P(s) = 0$   
 00100 => 3  $P(s) = 0.75$   
 10000 => 1  $P(s) = 0.25$



**POBLACION=4;**  
**GENERACIONES=2;**  
**UTILIDAD = {1, 2, 3, 4, 5};**  
**PESO = {1, 2, 3, 4, 5};**  
**PESO LIMITE = 12**

### Población Iteración 1:

- (0,0,0,1,0)
- (1,0,1,0,1)

### Población Iteración 0:

- (0,0,0,0,1) => 5  $P(s) = 0.38$
- (1,0,1,0,0) => 4  $P(s) = 0.31$
- (0,0,1,0,0) => 3  $P(s) = 0.23$
- (1,0,0,0,0) => 1  $P(s) = 0.08$



**POBLACION=4;**  
**GENERACIONES=2;**  
**UTILIDAD = {1, 2, 3, 4, 5};**  
**PESO = {1, 2, 3, 4, 5};**  
**PESO LIMITE = 12**

### Población Iteración 1:

- (0,0,0,1,0)
- (1,0,1,0,1)
- (0,0,0,0,0)
- (0,0,1,1,1)

### Población Iteración 0:

- (0,0,0,0,1) => 5  $P(s) = 0.38$
- (1,0,1,0,0) => 4  $P(s) = 0.31$
- (0,0,1,0,0) => 3  $P(s) = 0.23$
- (1,0,0,0,0) => 1  $P(s) = 0.08$



### Población Iteración 1:

- $(0,0,0,1,0) \Rightarrow 4$
- $(1,0,1,0,1) \Rightarrow 9$
- $(0,0,0,0,0) \Rightarrow 0$
- $(0,0,1,1,1) \Rightarrow 12$

Luego de 2 iteraciones encontramos una solución óptima al problema