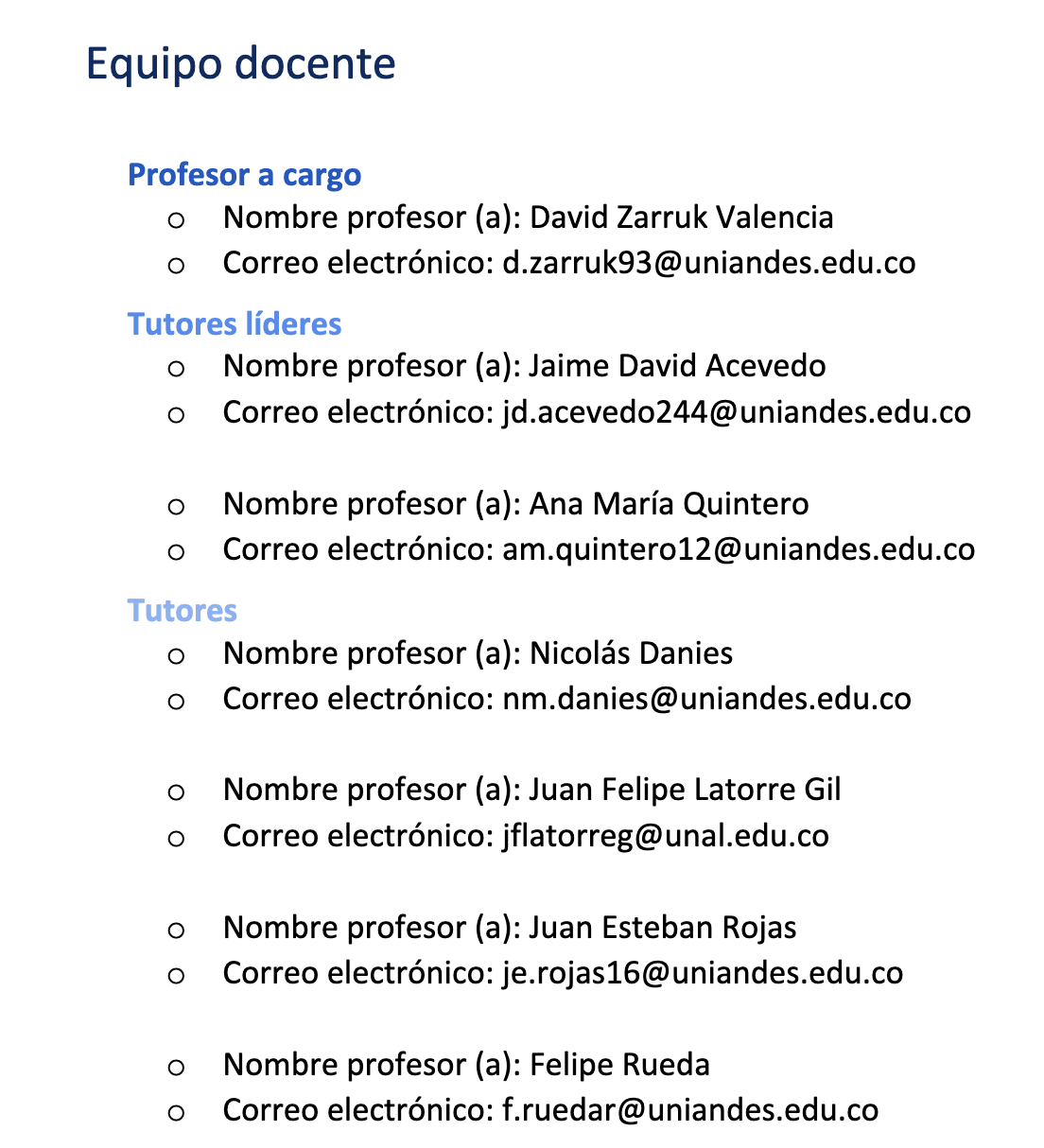
Machine Learning y Procesamiento de Lenguaje Natural

GitHub mio: [ocamilot/MIAD\_NLP\_2024: MIAD\_2024 (github.com)](https://github.com/ocamilot/MIAD_NLP_2024) (Fork del github del profe) 🡪 Aquí están todas las presentaciones, Jypiter Notebooks, etc.

GitHub profe: [davidzarruk/MIAD\_NLP\_2024 (github.com)](https://github.com/davidzarruk/MIAD_NLP_2024)







**Árboles de decisión**

<https://www.coursera.org/learn/machine-learning-procesamiento-lenguaje-natural/lecture/UCJbk/arboles-de-decision>

Bienvenidos a la sesión de árboles de decisión del curso Machine Learning y

procesamiento del lenguaje natural.

En la comunidad de Machine Learning,

los árboles de decisión son uno de los algoritmos

más usados para resolver diversos problemas.

Estos modelos tienen como objetivo predecir alguna

variable de interés a partir de reglas binarias sobre las variables predictoras.

Por ejemplo, en una planta de producción se quiere predecir el número de

piezas defectuosas a partir del tiempo que dura en la máquina 1 y en la máquina 2.

Se puede generar un árbol como este,

en el cual si el tiempo en la máquina 1 es menor a 9 minutos,

entonces se producen 10 piezas defectuosas.

Pero si el tiempo es mayor a 9 minutos y en la máquina 2 es mayor a 5 minutos,

entonces se obtienen 18 piezas defectuosas.

Por el contrario, en caso que en la máquina 2 se mueve menos de 5 minutos,

entonces se tendrán 5 piezas defectuosas.

Como podemos ver,

los árboles tienen diferentes componentes.

A la regla binaria del árbol se le conoce como "nodo interno",

en el que al cumplir una condición,

mediante la rama se llegará a otro nodo.

Los nodos que se encuentran al final del

árbol de los cuales no se desprende ninguna otra rama,

se les conoce como "nodos terminales" u "hojas".

Es acá donde se hará la predicción final.

Este tipo de modelo tiene varias ventajas, entre ellas,

que permiten una fácil interpretación de los resultados.

También que se pueden trabajar con variables categóricas y que no

se verán afectados por "oldies" ni por valores nulos en los datos.

También, puede ser usado tanto para

problemas de regresión como problemas de clasificación.

Los árboles de regresión son modelos en los que

se predice una variable de interés continua,

como por ejemplo, los niveles de colesterol,

el consumo de agua o el límite de crédito otorgado por un banco.

Dado que los árboles se crean a partir de reglas binarias,

hay dos opciones para su construcción.

La primera, en la que se realizan todas las posibles

particiones sobre las variables predictoras y la segunda,

en la que se realiza las particiones sobre las variables predictoras que permitan

una similaridad entre sus predicciones con el menor error en los nodos terminales.

¿Cuál de estas crees que es una mejor opción?

Aunque la primera parece correcta desde el punto de

vista teórico, es computacionalmente inviable.

Sería necesario una gran cantidad de evaluaciones para poder hacer una sola partición.

En cambio, la segunda

opción permite hacer la selección de variables de una

forma más inteligente para hacer las particiones.

Esta opción se conoce como "Binary Splitting".

Veamos la construcción de un árbol con esta metodología.

Retomando el problema anterior,

en la que queremos predecir el número de piezas defectuosas en una planta de producción,

primero en la parte superior del árbol,

en el nodo de origen se consideran todas

las posibles variables predictoras, el tiempo en máquina 1,

el tiempo en máquina 2 y vamos a agregar una tercera,

la cantidad de componentes para producir las piezas.

Además, para este ejemplo vamos a considerar 30 observaciones,

para las cuales se tiene un valor real del número de piezas

defectuosas y un valor pronosticado por medio del árbol.

Luego, para identificar qué variable va a ser usada en nuestro nodo de origen,

para cada una deberemos examinar todos los posibles puntos de corte.

En caso que tengamos una variable no numérica,

los puntos de corte son los diferentes valores categóricos que tiene esta variable.

Se hace la división o partición del nodo inicial en la variable y punto de corte,

para los cuales tenemos el menor error cuadrático medio;

es decir, la diferencia entre el valor real y el predicho sea lo más pequeño posible.

En el caso de nuestro ejemplo,

la primera opción sería el tiempo en la máquina 1,

que tiene como punto de corte utilizamos que sea mayor a 9 minutos; es decir,

después de esto vamos a partir el árbol en 2 utilizando esta nueva regla.

Por un lado, vamos a tener 19 observaciones que duraron más de 9 minutos.

De esta forma es que hacemos nuestra primera partición de datos.

Tras la partición, se evalúan ahora los dos nodos resultantes,

repitiendo los diferentes pasos.

Luego, se continúa con un proceso iterativo en el que se

repite el paso número cuatro hasta que lleguemos algún criterio de parada.

Los criterios comúnmente usados son la

profundidad máxima del árbol o el número mínimo

de observaciones que pueden haber en un nodo terminal o en una hoja.

En el caso, por ejemplo, si la

profundidad máxima del árbol que definimos es igual a 2,

tendríamos un total de 2 diferentes bifurcaciones.

Por otro lado, si para este ejemplo

el número mínimo observaciones en una hoja se establece como 3,

entonces se podría hacer otra división,

ya que en ninguno de los 2 nodos que tenemos,

tenemos menos de 3 observaciones.

Una vez se ha construido el árbol,

para predecir el valor de una nueva observación,

lo que tenemos que hacer es recorrer el árbol con sus diferentes

variables de esta observación hasta llegar a un nodo final.

De esta forma, utilizamos la información

que tenemos en estos nodos finales para hacer la predicción.

Por otra parte, los árboles de clasificación

siguen la misma lógica que los árboles de regresión,

solo que ahora el objetivo ya no es pronosticar una variable continua,

sino pronosticar una variable categórica.

Por ejemplo, si un usuario hace fraude o por ejemplo,

pronosticar cuál es el producto que un usuario más consume.

Además de la clara diferencia entre el tipo de variables

que predicen los árboles de regresión y de clasificación,

al construir el árbol de clasificación,

hay dos diferencias principales.

Primero, en el árbol de clasificación para predecir el valor de una hoja,

ya no promediamos los valores iniciales,

sino que miramos cuál es el valor más común.

A esto lo llamamos comúnmente como "votación".

Por otro lado, ahora ya no vamos a minimizar el error cuadrático medio,

sino vamos a minimizar algún otro tipo de métrica.

Veamos cómo podemos calcular algunas de estas métricas.

La primera sería la tasa de error de clasificación,

que se entiende como el porcentaje de

observaciones que no pertenecen a la clase más común del nodo.

Si tenemos un árbol en el que queremos predecir si un individuo compra el teléfono A o

el teléfono B y en cierto

nodo intermedio hay 10 individuos que compran el A y 15 que compran el B,

nuestra predicción es que todos comprarían el teléfono

B por ser la clase más común o la clase mayoritaria.

En este caso, la tasa de error sería del 40 por ciento,

ya que las observaciones de la clase minoritaria

se dividen por el total de las observaciones.

Ahora, si se hace una nueva regla

binaria por la variable predictora género, por ejemplo,

tendríamos 2 nuevos nodos en los cuales,

al cumplir la condición de ser hombre,

2 individuos comprarían el teléfono A,

y 2 el B,

mientras que si se cumple ser mujer,

8 individuos comprarían el teléfono A y 3

el B. El error de esta nueva predicción es del 20 por ciento,

pues se suma de las observaciones minoritarias 2 y 3,

divididas por el total el número de observaciones.

Ahora, el Gini Index es una medida de varianza total entre clases y se

calcula como 1 menos la sumatoria de las

proporciones de cada uno de los nodos elevados al cuadrado.

El máximo que puede tomar el Gini Index es de 0,5 y ocurre

cuando las clases están perfectamente balanceados en un nodo y el mínimo es 0,

que ocurre cuando solamente hay una clase en un nodo.

Siguiendo con el ejemplo anterior,

en el nodo en el que se tienen 10 individuos que comprarían el teléfono A y 15 el B,

el Gini Index sería el 0,48.

Para los nodos resultantes,

al dividir por género el Gini el de 0,24 y 0,40 para

tener un Gini final que se hace como la ponderación de estos diferentes Ginis.

De esta forma, tras la división se encuentra un Gini de 0,31.

Con esto concluimos el video de árboles de decisión.

Semana 4

**Introducción al Procesamiento de Lenguaje Natural**

[Introducción al Procesamiento de Lenguaje Natural | Coursera](https://www.coursera.org/learn/machine-learning-procesamiento-lenguaje-natural/lecture/PIbvq/introduccion-al-procesamiento-de-lenguaje-natural)

Bienvenidos a la sesión de Introducción al Procesamiento de Lenguaje Natural del curso.

Para entender el concepto de procesamiento del lenguaje natural,

primero, vamos a desglosar sus componentes.

El lenguaje se define como un sistema de signos o

estructuras que utilizan una comunidad para comunicarse dentro de un contexto;

es decir, es un sistema de comunicación que

permite transmitir ideas, sensaciones o sentimientos.

Ahora, ¿por qué es natural?

Lo natural hace referencia a la lengua o idioma hablado o escrito por humanos,

pues también existen otros tipos de lenguajes como el lenguaje animal,

matemático; por ejemplo,

cuando escribimos fórmulas y el lenguaje máquina,

que es cuando se escribe código en una computadora para que esta lo interprete.

Finalmente, ¿por qué se dice que se procesa el lenguaje natural?

Porque se busca que la máquina no solo capture el lenguaje humano,

sino que lo interprete,

reconozca, analice y genere.

Así, el procesamiento del lenguaje natural es una rama de la

inteligencia artificial que mediante diferentes técnicas y modelos,

busca darle sentido al lenguaje humano.

Continuemos con las principales

aplicaciones más conocidas de procesamiento del lenguaje natural.

"Recuperación de información" es encontrar resultados similares y relevantes,

tal y como lo hacen los motores de búsqueda de Google o Yahoo!.

"Extracción de información" es extraer

información de datos no estructurados, por ejemplo,

cuando en un correo se escribe, "nos vemos el sábado",

y el correo da la opción,

automáticamente, de agregar esto como un evento al calendario.

"Las traducciones automáticas" es pasar texto de un lenguaje

al otro y el "análisis de sentimientos",

que consiste en poder interpretar la actitud o sentimiento dentro de un texto.

Otra aplicación es la generación de lenguaje natural

en la que automáticamente se genera texto a partir de datos.

Por ejemplo, describir un partido a partir de las estadísticas de este.

También, y una de las aplicaciones más famosas,

es la de preguntas y respuestas automatizadas,

por ejemplo los "chatbots" de las páginas bancarias o de comercio.

Y, finalmente, la entrada de texto predictivo en la cual se sugieren palabras,

automáticamente, para terminar una oración,

como en nuestros celulares o motores de búsqueda.

Aunque el número de aplicaciones es grande no

significa que el procesamiento del lenguaje natural sea sencillo.

De hecho, dada la complejidad del lenguaje humano,

surgen diferentes retos,

entre ellos la ambigüedad, los modismos,

reconocimiento de entidades,

utilización de expresiones no formales y los recursos disponibles para el análisis.

Primero, "la ambigüedad",

que es cuando una palabra o expresión puede interpretarse de diferentes maneras,

es uno de los retos principales, es,

cómo hacemos para entender,

a la máquina, estas diferentes expresiones.

Por ejemplo, "la ambigüedad fonológica",

en la cual una cadena de sonidos puede ser confusa como "es conde" o "esconde".

Aunque las dos suenan similares,

la primera se refiere a que alguien tiene el título "de conde",

mientras que la segunda al verbo "esconder".

También tenemos la ambigüedad léxica,

en la cual una palabra tiene diferentes sentidos,

como la planta del pie o la planta del jardín.

Y la ambigüedad sintáptica,

donde, por ejemplo, se puede decir,

"Juan vio a Gabriel con sus gafas",

pero no es claro quién lleva las gafas, si Juan o Gabriel.

El siguiente reto son los modismos,

pues en el día a día tenemos diferentes expresiones,

no literales, para expresar ideas o sentimientos.

Por ejemplo, cuando se dice que alguien es la oveja negra,

se entiende como una persona rebelde,

o cuando se dice que alguien tiró la toalla es que alguien se dio por vencido.

¿Cómo hacemos que la máquina entienda esto?

El tercer reto es identificar entidades y es el problema de reconocer lugares,

nombres, animales y demás, dentro del texto.

El reto del español no estándar hace referencia

a poder entender las abreviaciones que usamos cuando escribimos,

por ejemplo, en mensaje de texto es común ver las palabras "pq" para decir "por qué",

o "bn" para decir "bien".

Finalmente, los recursos disponibles se refieren a

los recursos para el procesamiento del lenguaje natural,

los cuales dependen del idioma.

Por ejemplo, las librerías o herramientas que facilitan

el análisis de texto siguen siendo escasas en español.

El inglés es un idioma en donde se encuentran muchas más.

Con esto terminamos este video.

Nos vemos en una próxima oportunidad para

seguir profundizando las diferentes técnicas del

Reproduce el video desde :5:11 y sigue la transcripción5:11

procesamiento del lenguaje natural.

**Tokenización**

[Tokenización | Coursera](https://www.coursera.org/learn/machine-learning-procesamiento-lenguaje-natural/lecture/YbMmb/tokenizacion)

Bienvenidos a la sesión de "tokenización" del

curso "Machine Learning" y procesamiento del lenguaje natural.

La "tokenización" es un aspecto clave y obligatorio para procesar datos de texto.

Este proceso es una forma de separar

fragmentos de texto en unidades más pequeñas llamadas "tokens",

las cuales ayudan a comprender el contexto y a

interpretar su significado al analizar la secuencia de "tokens".

De manera general, existen tres tipos de

"tokenización": la "tokenización" de palabras,

de caracteres y de "n-gramas".

Primero, la "tokenización" de palabras consiste en

dividir una oración en el conjunto de palabras que la componen.

Por ejemplo, si hacemos "tokenización" de palabras

de la frase "hamburguesa con carne a la parrilla",

tendríamos seis "tokens" diferentes: "hamburguesa",

"con", "carne", "a", "la" y "parrilla".

Esta misma frase también la podríamos descomponer en "tokens" de caracteres, es decir,

que se divide la oración por cada carácter o letra que la componen,

lo que nos daría para este ejemplo 30 tokens: "H",

"a", "m", etc.

Ahora, la totalización

de "n-gramas" consiste en descomponer la frase en conjuntos de palabras.

Al "tokenizar" por "n-gramas" de una palabra,

tendríamos un "unigrama",

el cual estaría constituido

por los mismos "tokens" que se obtienen al hacer una "tokenización" por palabras,

tal y como estaba en nuestro ejemplo anterior.

Si "tokenizamos" por "n-gramas" de dos palabras,

tendríamos "bigramas" de cinco "tokens": "hamburguesa con",

"con carne" y así sucesivamente.

El proceso de "tokenización" es muy importante para utilizar

los principales algoritmos de procesamiento de lenguaje natural.

Sin embargo, dada la complejidad del lenguaje humano,

hay diferentes retos que se deben tener en

cuenta para mejorar el procesamiento de la información.

Primero, hay palabras que juntas tienen un

significado distinto a las palabras por separado.

Por ejemplo, con la palabra "San Francisco" como ciudad,

se debe decidir si se debe fragmentar en uno o en dos "tokens".

Por otra parte, por ejemplo, en el idioma inglés,

el apóstrofe posesivo hace parte de la gramática,

por lo que es necesario decidir si en la asignación de "tokens",

el apóstrofe se debe considerar o no.

Todo para tener una homogeneidad en la "tokenización".

En el caso posesivo de la capital de Finlandia,

que en inglés se escribe como "Finland" apóstrofe "s",

se debe decidir si se quita el apóstrofe,

la "s", o si ambos se dejan para constituir uno o dos "tokens".

Otro reto está dado por las palabras compuestas separadas por un guión,

como por ejemplo "franco-germánico".

Acá debemos decidir,

al hacer la "tokenización" de caracteres,

si el guión cuenta o no como un "token" y al hacer la "tokenización" de palabras,

si vamos a considerar uno o dos "tokens".

En idiomas de origen romance,

como el francés y el italiano,

que tienen reglas gramaticales las cuales colocan apóstrofes para eliminar vocales,

también se generan cuestionamientos de cómo se debe hacer la "tokenización".

Por ejemplo, en francés,

los artículos "Le", "L",

cuando van acompañados de la palabra que inicia con una vocal,

se reducen a apóstrofe "L",

en el caso del artículo "Le" cuando va acompañado de la palabra "ensemble".

Por lo tanto, si se quiere hacer "tokenización" de palabras,

se debe decidir si se cuenta o no la palabra

completa que podría o no contener el apóstrofe,

o si son dos, o el artículo reducido y la palabra,

o el artículo completo y la palabra.

Como hemos visto, varios de estos retos surgen por la complejidad de los idiomas.

Ahora, analizaremos los retos que obtenemos en idiomas del oriente,

como sería el chino o el japonés.

En estos dos idiomas no se hace uso de

espacios y las palabras se componen de diferentes símbolos,

lo que hace que el proceso de "tokenización" sea aún

más complejo y esto disminuye la precisión de los resultados.

Por ejemplo, la frase "Sharapova ahora vive en US sureste Florida".

Cada palabra en chino es un conjunto de símbolos,

por lo que cada uno de estos conjuntos debería ser considerado como un "token".

Adicionalmente, en el japonés,

diferentes alfabetos son mezclados,

como por ejemplo el "katakana" y el "kanji",

lo que hace el proceso aún más complejo.

En general, la decisión para cada uno de estos retos depende del problema de

procesamiento de lenguaje natural que se esté estudiando y del lenguaje usado.

Con esto terminamos este vídeo.

Reproduce el video desde :4:36 y sigue la transcripción4:36

Nos vemos en una próxima oportunidad.

**N-Gramas**

[N-Gramas | Coursera](https://www.coursera.org/learn/machine-learning-procesamiento-lenguaje-natural/lecture/TEYxf/n-gramas)

Bienvenidos a la sección de N Gramas del curso machine

Learning y procesamiento de lenguaje natural.

Reproduce el video desde ::16 y sigue la transcripción0:16

Los n gramas son un conjunto de n elementos consecutivos en una

frase u oración.

Que son usados para diferentes aplicaciones predictivas del

procesamiento de lenguaje natural.

Reproduce el video desde ::29 y sigue la transcripción0:29

Por ejemplo, la frase necesito ir al banco a pagar mis deudas

se puede descomponer en 7. Digamos que son conjunto de 2

palabras de largo. También se puede descomponer en trigramas,

conjunto de 3 palabras, teniendo un total de 6 programas como

necesito ir al.

¿O banco para pagar?

Reproduce el video desde ::48 y sigue la transcripción0:48

Dogmas son importantes para resolver diferentes problemas de

predicción en los que necesitamos y evaluar qué tan

correcto o coherente es un conjunto de palabras.

Por ejemplo, en un problema de entrada de texto predictivo. ¿Si

alguien escribe en su celular nos vemos que es sugerencia, es

más coherente, nos vemos pronto o nos vemos rápido?

Reproduce el video desde :1:10 y sigue la transcripción1:10

Para un humano es fácilmente identificable que es más común,

nos vemos pronto, pero para una máquina o modelo que haga esta

tarea debemos usar modelos de lenguaje probabilísticos YN

gramas.

Reproduce el video desde :1:23 y sigue la transcripción1:23

Estos modelos tienen como objetivo estimar la probabilidad

de ocurrencia de una oración o secuencia de palabras.

Para el caso del ejemplo que acabamos de ver, nos vemos

pronto debería tener una probabilidad mayor de

ocurrencia.

Reproduce el video desde :1:36 y sigue la transcripción1:36

Por lo que el modelo debe sugerir la palabra pronto.

Reproduce el video desde :1:40 y sigue la transcripción1:40

En general, estos modelos básicamente nos permiten

calcular la probabilidad de una secuencia de palabras W.

Esta probabilidad se define como la probabilidad de cada una de

las palabras wb y que componen la secuencia o frase.

La probabilidad de cada una de las palabras se puede definir

como la probabilidad condicional de la palabra wb, y dado. N

palabras anteriores, por ejemplo, la probabilidad

condicional en la frase Te quiero la palabra que sigue.

Más probable es la palabra mucho, pues es mucho más común

que, por ejemplo, la palabra bastante.

Reproduce el video desde :2:17 y sigue la transcripción2:17

Retomando la probabilidad de ocurrencia de las secuencias de

palabras, se definen con entonces como el producto de la

probabilidad condicional. La I ésima palabra dada las palabras

anteriores de la uno a la I sub uno.

Reproduce el video desde :2:31 y sigue la transcripción2:31

Por ejemplo, para la frase la casa es muy grande, la

probabilidad de que esta frase ocurra es la probabilidad de la

palabra la multiplicado por la probabilidad de la palabra casa

dado la y así sucesivamente.

Cuando las frases son muy largas para el cálculo, para

simplificar este cálculo utilizamos el supuesto de

Markov.

El supuesto Markov indica que para calcular la probabilidad

basta con considerar solamente las n anteriores, continuando

con nuestro ejemplo, la frase la casa es muy grande, si queremos

predecir la palabra querida después de la casa es muy grande

y entonces, según Marcos, van basta con verlas. N anteriores.

Por ejemplo, si queremos saber la probabilidad de que la

palabra que sigue es bonita, te puede calcular como la

probabilidad de la palabra bonita, dado que la palabra

anterior es I.

¿O también se puede tener en cuenta las 2 anteriores como un

grande?

Como podemos ver, el supuesto de Markov, Ceuta n Gramas y la

probabilidad de ocurrencia de una palabra se definirá

formalmente como el producto de la probabilidad de la I ésima

palabra dado las. N palabras anteriores, hablemos ahora de

cada uno de los modelos de lenguaje probabilísticos, el más

simple es el modelo de una gama donde la probabilidad de

ocurrencia de una secuencia es el producto de las

probabilidades independientes de cada palabra.

Reproduce el video desde :3:50 y sigue la transcripción3:50

Esto significa que el modelo genera oraciones seleccionando

aleatoriamente cada palabra sin ningún sentido o contexto entre

ellas, por ejemplo, una lista de palabras como al perro ande,

caza, corre.

Reproduce el video desde :4:4 y sigue la transcripción4:04

Para no predecir cualquier grupo de palabras al azar, el modelo

VIP de big Drama calcula la probabilidad de cada palabra,

dado la palabra inmediatamente anterior. Por ejemplo, un modelo

de enigma puede generar la siguiente frase, cargador,

carro, rueda, calle, pavimento material osteológico, animales,

perros.

Reproduce el video desde :4:23 y sigue la transcripción4:23

Aunque las palabras de las parejas tienen relación, la

frase todavía no termina teniendo coherencia.

¿Finalmente, un modelo de n grama para predecir la palabra

que mejor complemento, una frase debe considerar, cuál es el

número n necesario para capturar contexto y que se ajuste?

Para capturar la frase estoy muy cansado, debería sentarme en una

basta con considerar el trigrama sentarme en una para identificar

que la palabra que sigue es un elemento que permite permite

asentarse como una silla.

Por otra parte, si la frase Tengo mucha hambre, deberíamos

ir aún, entonces, no basta con 3, debemos considerar las 5

anteriores, pues hambre es una palabra que da contexto y

permite identificar una buena palabra de de un restaurante y

no parqueadero por, ejem.

Reproduce el video desde :5:12 y sigue la transcripción5:12

Por último, veamos un ejemplo de cómo se calculan las

probabilidades, las probabilidades se calculan como

la ocurrencia en un texto o en diferentes oración.

Por lo que, considerando únicamente estas 3 frases, si se

quiere calcular la probabilidad de que después de la palabra me

siga la palabra gusta, entonces debemos identificar la cantidad

de veces que esto ocurre y dividirlo por el total. En el

caso de me gusta, en esta hora está en la oración uno y 2, por

lo que una prioridad de 2/3. Por otra parte, la probabilidad de

que la ocurrencia de la palabra soy después de la palabra yo

solamente de 1/3. Por último, la probabilidad de ocurrencia del

birama gusta comer es cero, pues en este caso no, no está en

ninguna de las oraciones anteriores.

Reproduce el video desde :5:59 y sigue la transcripción5:59

Con esto terminamos este vídeo, nos vemos en una próxima

oportunidad.

Semana 5:

**Similitud de textos**

[Similitud de textos | Coursera](https://www.coursera.org/learn/machine-learning-procesamiento-lenguaje-natural/lecture/AK9In/similitud-de-textos)

[MÚSICA]

[MÚSICA] Bienvenidos a la sesión de similitud

de texto del curso matching learning y procesamiento de lenguaje natural,

Reproduce el video desde ::16 y sigue la transcripción0:16

La técnica de similitud de textos en el procesamiento de lenguaje natural,

busca estimar el grado de similitud entre dos textos, esto, de una manera eficiente

por ejemplo, en los motores de búsqueda es necesario comparar el contenido de las

páginas disponibles con la frase buscada, con el fin de encontrar aquellas

páginas que contengan la misma frase o una similitud semántica.

Reproduce el video desde ::40 y sigue la transcripción0:40

Otro ejemplo, son los chatbots, los cuales deben ser capaces de dar respuestas

uniformes u unánimes a la consulta o preguntas semánticamente similar.

Reproduce el video desde ::50 y sigue la transcripción0:50

Existen diferentes técnicas para calcular qué tan parecidos son dos textos,

hoy veremos tres de ellas, similitud de Jaccard,

similitud de coseno y codificación de oraciones.

Reproduce el video desde :1:1 y sigue la transcripción1:01

Primero, la similitud de Jaccard calcula la proporción de palabras que se

interceptan o tienen en común dos textos,

esto se calcula como la razón entre la cardenalidad de la intersección de las

palabras y la cardinalidad de la unión de los dos textos.

Por ejemplo, si se tiene la frase A, la inteligencia artificial ha mejorado muchos

problemas, y la frase B, muchos problemas se resuelven con inteligencia artificial,

entonces la similitud de Jaccard es de 0.4.

Pues, las frases tienen en común cuatro palabras

y hay un total de palabras en las dos oraciones de diez.

[MÚSICA] Continuando con las diferentes técnicas, la similitud de

coseno permite calcular qué tan parecidos son dos textos, pero esta vez calculando

el coseno del ángulo entre los dos vectores que representan los dos textos.

Esta similitud se calcula con el producto de los vectores dividido

por el producto de las normas de cada vector.

Esto quiere decir que si el coseno es de 1, entonces los dos vectores son iguales,

si es de 0, son totalmente diferentes, y menos 1 es que son opuestos.

Retomando el ejemplo anterior, primero se debe armar una lista con las palabras

únicas de los dos textos, en este ejemplo, son diez, luego, con base en esta

lista de palabras, se elabora un vector binario para cada uno de los dos textos,

que especifique la presencia o no, de cada una de las palabras de la lista.

1 sería que sí está, 0 que no está.

En el caso del texto A, el primer componente del vector es 1,

pues el texto contiene la primera palabra de la lista, o sea, la palabra artificial.

En el segundo, componente 0,

pues no contiene la segunda palabra que en este caso es la palabra con.

Una vez se tienen los vectores, se hace el producto punto entre ellos

y se divide por el producto punto de las normas tal y cual se indicó anteriormente.

Para nuestro ejemplo, el producto punto es 4 y el producto punto de las normas es 7,

lo que hace que la similitud sea de 0.57.

[MÚSICA] Por último,

la codificación de oraciones o sentence embedding, como se conoce en inglés,

es una técnica en la cual los dos textos se representan con vectores numéricos.

Al poder representar numéricamente las frases o textos,

al final de esta técnica se hace uso nuevamente de la similitud del coseno

para cuantificar qué tan similares son.

En la codificación de oraciones, you no es necesario generar un vector binario,

sino que ahora generamos un vector con coordenadas que logren capturar el

contexto de una palabra en el documento en relación con las otras palabras.

Existen diferentes metodologías para el proceso de codificación de oraciones,

unas de las más conocidas son Doc2Vec,

para español, y SentenceBert, InferSent, para textos en inglés.

Si bien no vamos a profundizar en el proceso de codificación de cada una de

ellas, pues no hacen parte de este curso, es importante que las conozcan.

En general, independientemente del proceso de codificación de oraciones que se use,

se generan vectores numéricos tal cual se muestra en la imagen, las frases similares

van a quedar en espacios similares, mientras que las frases que tienen

significados completamente diferentes van a quedar en espacios alejados.

Los valores en cada uno de estos vectores son el

resultado del proceso de codificación que se haya decidido usar.

Con estas representaciones se puede hacer la similitud del coseno,

y obtener una medida de similitud más robusta

que solamente considerar el vector binario de ocurrencia de palabras.

[MÚSICA] Con esto terminamos este video.

Nos vemos en una próxima oportunidad.

MÚSICA]

**Normalización y Stopwords**

[Normalización y Stopwords | Coursera](https://www.coursera.org/learn/machine-learning-procesamiento-lenguaje-natural/lecture/vob03/normalizacion-y-stopwords)

Bienvenidos a la sesión de normalización y stopwords del curso Machine Learning

y Procesamiento del lenguaje natural.

La normalización es el proceso de convertir una palabra en su forma más básica.

Esto quiere decir que se busca reducir los cambios o derivaciones

de una palabra a una forma más común,

cuando las variaciones tienen el mismo significado.

Hay dos métodos muy populares en

la comunidad machine learning para hacerlo: stemming y lematización.

El stemming es un proceso en el que

para cada palabra se obtiene su raíz o stem en inglés.

Este es la unidad central portadora

de significado de la palabra y es compartido

por las distintas variantes e inflexiones de una palabra.

Por ejemplo, para la palabra florido,

florero, florista y floral, su raíz es flor.

El uso de este método facilita el procesamiento de textos.

De manera general, existen diferentes algoritmos

para obtener los stems de las palabras de un texto.

Para el español esos algoritmos se basan en

reglas que remueven de las palabras las terminaciones verbales,

plurales o de género,

como es el caso del algoritmo SnowballStemmer de la librería NLTK.

Por ejemplo, para la frase "Pablo jugaba en la arenera" el stem se ve esta forma.

En inglés el algoritmo más usado es el algoritmo de Porter´s,

el cual también se basa en un robusto set de reglas gramaticales en las que,

por ejemplo, la palabra cats se transforma en cat.

El proceso de stemming tiene dos desventajas principales.

Primero, que puede generar las mismas

raíces para dos palabras con significados diferentes.

Y segundo, que puede generar raíces que carecen de significado en el texto.

Por ejemplo, para las palabras niños y niñas,

la raíz es el mismo,

aun cuando las palabras son diferentes.

Además, esta raíz no tiene ningún significado por sí solo,

ya que la palabra no existe en español.

Para evitar estos problemas,

el método de lematización,

o lemmatization en inglés,

ofrece un proceso más robusto en el que se busca el lema de cada palabra.

El lema se define como la forma base o de diccionario.

Por ejemplo, para las palabras dije,

diré, dijéramos el lema es decir.

Además, otra ventaja es

que el lema resultante también considera el contexto de la palabra.

Por ejemplo, para la palabra traje,

de acuerdo con las palabras predecesoras,

el algoritmo es capaz de identificar si el lema

referente al verbo traer o al sustantivo traje, de ropa.

Dado que los lemas capturan una mayor complejidad,

los algoritmos de lematización también se vuelven más complejos.

Dado el alcance del curso,

no entraremos en el detalle de estos algoritmos.

Sin embargo, es importante saber que estos realizan

un etiquetado gramatical o de "Part Of Speech" en inglés,

es decir, que le asignan a cada palabra la categoría gramatical a la que pertenecen,

lo que permite retornar un lema con significado y contexto.

Para nuestro ejemplo anterior,

Pablo jugaba en la arenera,

los algoritmos de lematización le asignarán la

categoría sustantiva a las palabras Pablo,

la, arenera, mientras que le asignarán la categoría verbo a la palabra jugar.

Por último, otro procesamiento básico de texto es la eliminación de stopwords.

Las stopwords son las palabras más comunes en cualquier

lenguaje natural y que puede que no agreguen valor al significado del texto.

Por ejemplo, en español,

algunas de ellas son: y,

al, con, de, entre otras.

Dado que estas palabras no generan valor al texto,

el eliminarlas puede mejorar el rendimiento de nuestros modelos proyectivos.

Con esto terminamos este video.

Nos vemos en una próxima oportunidad.

SEMANA 7

**Redes neuronales recurrentes**

[Redes neuronales recurrentes | Coursera](https://www.coursera.org/learn/machine-learning-procesamiento-lenguaje-natural/lecture/dRav3/redes-neuronales-recurrentes)

Bienvenidos a la sesión de "Redes neuronales recurrentes",

del curso "Machine learning y procesamiento del lenguaje natural".

Las redes que se han visto hasta el momento en el curso,

están diseñadas para variables de entrada y predicciones no secuenciales.

Por ejemplo, para un problema en el que se quiere

predecir si un individuo va a sufrir de una enfermedad cardiaca,

se usan variables como la presión arterial y el peso, tomados a la fecha.

Sin embargo, para ese problema y muchos otros,

el historial o secuencia de tiempo de estas variables puede ser importante,

pues pueden dar información de eventos que se entrelazan en el tiempo.

Por facilidad de representación de la red,

para esta lección vamos a representar la capa

de entrada como una sola neurona de color azul;

la capa oculta, con una caja de color verde y la capa de salida,

como una sola neurona de color morado.

Estas redes, al no tener en cuenta el comportamiento de las variables en tiempo,

están limitadas, no pueden aprender de los eventos anteriores,

porque la información no se puede transferir en el tiempo.

Para nuestro ejemplo de enfermedad cardíaca,

puede ser importante identificar el comportamiento

de las diferentes variables a lo largo del tiempo,

como es el caso de la presión arterial.

Si una persona ha venido teniendo una presión elevada en los últimos cuatro meses,

puede que sea más propensa

a tener una enfermedad cardíaca que alguien que la ha tenido estable.

Con las redes vistas,

no es posible capturar la secuencia de esta información, ya que,

para su predicción de cada mes se utilizaría

una red independiente con datos en el tiempo t,

sin ninguna relación con períodos anteriores.

Para resolver ese tipo de problemas,

se desarrollan las redes neuronales

recurrentes o "Recurrent Neural Networks", en inglés.

Estas son un tipo de red neuronales para datos secuenciales,

es decir, que se enlazan con el paso en el tiempo.

Por ejemplo, el precio de las acciones del mercado bursátil,

los datos recolectados por un sensor o texto.

Específicamente, en el caso de los textos,

estos se pueden dividir en una secuencia de caracteres o palabras donde

cada una se puede pensar en un momento t en una secuencia de largo T mayúscula.

En el caso de la frase "Me gusta machine learning",

la palabra "Me",

está en el momento t igual a cero, y así, sucesivamente.

Para poder transmitir información,

estas redes incluyen una memoria interna o estado interno llamado "h\_t",

conocidos en inglés como "Internal Memory" o "Internal State",

el cual captura información en el momento t y lo transmite al momento t más 1.

De esta forma, las redes

neuronales recurrentes no solo consideran las variables de entrada en el tiempo t,

sino que tienen en cuenta también las del estado anterior, t menos 1.

Veamos ahora, los diferentes casos de aplicación de las redes neuronales recurrentes.

Primero, se tiene el caso,

"Many to One" o "muchos a uno",

en el cual, se tiene varias entradas secuenciales y un solo resultado o predicción.

Por ejemplo, para el análisis de sentimientos,

se recibe una cadena de texto,

pero se quiere devolver solamente el sentimiento asociado con toda la cadena.

Luego, tenemos el caso "One to Many" o "uno muchos",

en el que se tiene una sola entrada,

pero múltiples predicciones secuenciales.

Este es el caso de generación de música,

en las que se toma una nota musical y se produce una secuencia o canción.

Por último, en el caso de "Many to many" o "muchos a muchos",

se tienen múltiples entradas y resultados secuenciales.

Este es el caso de la traducción automática,

en el que se tiene un texto y una palabra que debe ser traducida.

Es importante tener en cuenta que existen varios tipos de redes neuronales recurrentes.

De estas, las más usadas por la comunidad de machine learning son las redes

neuronales de memoria a corto y largo plazo o el STM, por sus siglas en inglés.

Las cuales se caracterizan por ser capaces

de aprender dependencias tanto de corto como de largo plazo,

es decir, que guardan información de estados

previos que puede ser relevante para las proyecciones en futuros estados.

Para guardar información o tener memoria,

las LSTM tienen estructuras llamadas,

"compuertas" o "gates",

que permiten agregar o remover información.

Por ejemplo, una compuerta con una función

sigmoide hace que toda la información que pase a través de ella quede en el rango 0 o 1.

Donde 0 quiere decir,

que no se pasa ninguna información y 1,

que se pasa la información completa.

Para entender con más detalle cómo funcionan las LSTM,

debemos considerar que su funcionamiento se da gracias a una compuerta de olvido,

una compuerta de entrada,

una de actualización de estado y una compuerta de resultado.

Veamos ahora, en qué consiste cada una de ellas.

Primero, la compuerta de olvido o "Forget Gate", en inglés,

decide qué información del estado h\_t menos 1,

se debe olvidar, es decir,

qué información no es relevante.

Esto se realiza con una función sigmoide en la que,

como ya mencionamos antes,

un 1 representa que la información se debe mantener por completo y 0, lo contrario.

Luego, la compuerta de entrada o "Input Gates",

decide que información nueva se debe guardar en el estado de la celda,

a partir de tres etapas.

La primera, ocurre con una capa llamada "la capa de puerta de entrada" o "i\_t",

donde se decide qué valores actualizar con una función sigmoide.

La segunda, ocurre en una capa llamada "la capa tangente hiperbólica",

donde se genera un vector con los nuevos valores

candidatos que podrían agregarse al estado de la celda.

Y la tercera ocurre cuando se pondera la información de las dos capas anteriores.

El tercer paso es actualizar el estado de la celda.

Este se hace a partir del estado de la celda anterior,

ponderado por el resultado de la compuerta de

olvido y sumando el resultado de la compuerta de entrada.

Es en este paso donde se combina la información pasada con la actual.

Y por último, la compuerta de resultado,

se retorna el estado interno h\_t,

y la proyección y\_t.

Con esto terminamos este video.

Nos vemos en una próxima oportunidad.

**Transfer Learning**

[Transfer Learning | Coursera](https://www.coursera.org/learn/machine-learning-procesamiento-lenguaje-natural/lecture/LMEzZ/transfer-learning)

Bienvenidos a la sesión de Transfer Learning

del curso "Machine Learning y Procesamiento de Lenguaje Natural".

El aprendizaje por transferencia es un método de aprendizaje automático en el que

se reutiliza un modelo ya entrenado para la proyección de otra tarea similar.

Es decir, se busca aprovechar lo aprendido en

una tarea para mejorar la generalización en otra.

El aprendizaje por transferencia

se utiliza principalmente en tareas de procesamiento de lenguaje natural y de VIS,

de visión artificial o "computer vision",

el cual se enfoca en el procesamiento y el análisis de imágenes.

Lo anterior, debido a que para esos

problemas se requieren de muchos datos para entrenar los modelos,

además de una alta capacidad computacional.

De esta forma, en lugar de comenzar el proceso de aprendizaje desde cero,

se inicia con patrones aprendidos al resolver una tarea relacionada.

Las principales ventajas de este método de

aprendizaje son que ahorra tiempo de entrenamiento;

que no necesita de muchos datos y suele dar

mejor rendimiento con respecto a modelos que hayan sido entrenados desde cero,

con pocos datos y con poca capacidad computacional.

El tiempo de entrenamiento se reduce,

pues para tareas complejas,

entrenar una red neuronal puede tardar días, incluso semanas.

Tampoco se requieren muchas observaciones,

debido a que los modelos que se reutilizan ya han sido entrenados previamente.

Esto es de gran beneficio,

ya que generalmente el acceso a un gran número de datos no es siempre factible.

Y por último, se tiene un mejor rendimiento,

dado que los modelos que se reutilizan son modelos robustos,

que también han sido entrenados con muchos datos.

Con base en lo anterior, el aprendizaje por transferencia

se puede usar cuando no hay suficientes datos

de entrenamiento etiquetados para entrenar un modelo desde cero

y cuando ya existe un modelo previamente entrenado en una tarea similar,

lo que implica que haya sido entrenado con un alto número de datos.

Existen muchos modelos preentrenados,

de los cuales solo describiremos,

de manera general, los más usados para problemas de

procesamiento del lenguaje natural y de "computer vision".

Primero, para procesamiento del lenguaje natural,

cinco de los más populares son: OpenAl's GPT-3,

Google's BERT, Microsoft's CodeBERT, ELMo y XLNet.

En específico, OpenAl's

GPT-3 es un modelo de lenguaje entrenado con 175.000 millones de parámetros,

lo que ha permitido tener un comportamiento robusto en tareas como traducción,

responder preguntas y descifrar palabras.

De hecho, ha sido utilizado, incluso,

para escribir artículos de noticias y para generar código que ayudan a desarrolladores.

Por otra parte, para problemas de "computer vision" se cuenta con modelos como:

VGG-16, ResNet50, Inceptionv3, y EfficientNet,

que, de manera general, son modelos preentrenados para clasificación de imágenes.

Particularmente, VGG-16 es uno de los más

robustos con más de 138 billones de parámetros de entrenamiento.

Veamos ahora cómo funciona el aprendizaje por transferencia.

Este, se rige en tres pasos principales.

Primero, se debe seleccionar el modelo preentrenado.

Luego, este se debe adaptar para crear un nuevo modelo y finalmente,

se entrena solo el nuevo modelo adaptado.

Veamos esto con más detalle.

Supongamos que se quiera resolver una tarea de

clasificación de género a partir de imágenes,

es decir, si un individuo es hombre o mujer, dado una foto.

Primero, se selecciona el modelo

preentrenado para la detección de rostros, por ejemplo,

VGGFace2 o FaceNet,

los cuales han sido entrenados con millones de imágenes,

lo que les permite detectar rostros en una nueva imagen.

Ahora, para adaptar el modelo preentrenado,

al problema de clasificación por género,

se cambia la estructura de la red,

de forma que se adapte a la nueva tarea requerida.

En este caso, dado que se requiere

clasificar el rostros de una imagen como hombre o mujer,

se dejan dos neuronas en la capa de salida.

Por último, se entrena el modelo adoptado.

Esto depende de la cantidad de observaciones disponibles.

Por un lado, si se tienen pocas observaciones,

se pueden entrenar los pesos o parámetros de la última capa.

Por otra parte, si la cantidad de datos es considerable,

entonces se puede entrenar, incluso,

las capas ocultas anteriores.

Es importante aclarar, si se tiene un dataset muy grande,

el modelo se puede entrenar por completo,

lo cual se conoce como "fine-tunning".

Reproduce el video desde :4:56 y sigue la transcripción4:56

Con eso, terminamos este video.

Nos vemos en una próxima oportunidad.