OBS Business School

RED DE EDUCACIÓN SUPERIOR
Planeta Formación y Universidades

Jerónimo Molina Molina

Procesamiento del Lenguaje Natural

Introducción

Introducción

Bienvenido a la asignatura de Procesamiento del Lenguaje Natural. En el recorrido que vamos a hacer juntos de esta materia intentaré, por un lado, que conozcas cómo ha evolucionado la misma, destacando que es el primera área de trabajo que surgió en el campo de la inteligencia artificial, antes incluso de la conferencia de Darmouth. Por otro lado, trataré de presentar las últimas arquitecturas y los conceptos más relevantes que aún hoy tienen vigencia y se aplican en proyectos reales.

Pero, antes de meternos en faena, quiero definir una serie de objetivos concretos, que deberás tener en cuenta para, alcanzándolos, sacar el máximo beneficio de esta asignatura.

Objetivos

1. Abordar en profundidad, y no solo desde una perspectiva técnica sino también de negocio el procesamiento del lenguaje natural.
2. Conocer sus orígenes, y su evolución, desde las técnicas más clásicas (y no tan obsoletas) como la tendencia actual, marcada por los LLM (Large Language Models)
3. Entender las diferentes tareas que se abordan en NLP y su aplicación.
4. Conocer, como base de las arquitecturas actuales los Embeddings y comprender cómo permiten establecer relaciones entre tokens. Conocer cómo desde Python se pueden emplear distintas librerías para realizar tareas básicas de NLP.
5. Conocer cómo puede entrenarse un modelo transformer desde cero, para entender el proceso y su funcionamiento.
6. Conocer diferentes repositorios de modelos y de herramientas NLP que pueden utilizarse en línea para desarrollar tareas avanzadas de NLP.
7. Entender en qué consisten los LLM, asimilarlos como estado del arte actual del PLN y entender las posibilidades que brindan, on cloud y on premise.

El Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural (NLP) es una rama de la inteligencia artificial (IA) que permite a las computadoras comprender, generar y manipular el lenguaje humano, por lo que tiene una componente lingüística muy alta, en un intento de procesar la información de un modo similar al que empleamos los humanos para analizarla (con independencia de cómo lo haga nuestro cerebro, del que, dicho sea de paso, se conoce muy poco todavía).

Orígenes del P.L.N y el Test de Touring

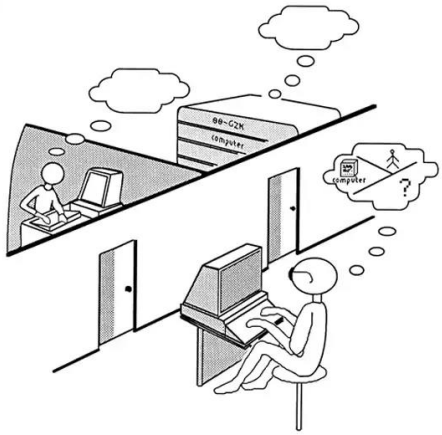
Esta disciplina data de los años 50, cuando Alan Turing publicó un célebre artículo titulado “Computing Machinery and Intelligence” (https://redirect.cs.umbc.edu/courses/471/papers/turing.pdf), por lo que nos

situamos mucho antes de que en Darmouth se acuñara el término “Inteligencia Artificial” como el área de conocimiento (hoy ciencia) que engloba a esta y a otras líneas de trabajo.

La repercusión de este artículo de investigación alcanza hasta nuestros días desde diferentes perspectivas, pero si por algo es conocido es por el planteamiento de lo que se conoce como “Test de Turing” que tardó muchos años en superarse y que a día de hoy, aunque superado por muchos sistemas de información, sigue siendo una referencia. Si sientes curiosidad de invito a que leas no solo el artículo mencionado, sino también su predecesor, titulado “Intelligent Machinery” (https://www.npl.co.uk/getattachment/about-us/History/Famous-faces/Alan-Turing/80916595-Intelligent-Machinery.pdf?lang=en-GB). En este artículo se basan las famosísimas “máquinas de Turing”, un modelo matemático original que permite simular cualquier sistema computacional. Si deseas conocer algo más sobre la historia de Alan Turing, te sugiero que veas la película titulada “Descifrando a Enigma”.

El Test de Touring

Este test se da en un escenario en el que hay, en tres habitaciones separadas, dos personas (A, B) y una máquina (M), que es una inteligencia artificial. La persona A debe comunicarse telemáticamente (por ejemplo vía chat) con (B) y con (M), siedo el objetivo de este experimento determinar si (M) puede imitar las respuestas humanas hasta el punto de conseguir que (A) no sepa diferenciar entre esta máquina (M) y la otra persona (B).



Fuente: hipertextual.com

Para ello, el humano (A) hace preguntas tanto a la otra persona (B) como a la inteligencia artificial (actualmente, por lo general, un chatbot) y si no puede identificar si alguno de los dos sujetos es o no una máquina, la IA pasa con éxito “El Test de Turing”.

El Test de Touring 2.0

Precisamente en torno a la fecha de elaboración de este contenido, 22 de junio de 2023 sale a la luz la propuesta que Mustafá Suleyman, fundador de DeepMind, hace en su libro “The Coming Wave: Technology, Power, and the Greatest Dilemma of the Twenty-First Century” (que en breve estará disponible en su “librería favorita…” -2023/09-).

Se trata de una propuesta que, más allá de la de Turing, pretende discernir cuándo una máquina califica como “Inteligencia Artificial capaz”. En concreto, la capacidad de realizar de forma muy sencilla tareas que

a un humano promedio le suponen un esfuerzo considerable. Por ejemplo, desarrollar un negocio con 100.000€ y generar 1.000.000€… lograr este tipo de retos es lo que, según Suleyman, califica, a una IA, como una "IAC".

Puedes documentarte un poco más sobre este hito en:

- El nuevo libro de Suleyman, que cuando leas esto, probablemente ya esté publicado.
- https://hipertextual.com/2023/06/asi-es-el-nuevo-test-que-evalua-la-capacidad-de-la-inteligencia-artificial-para-ganar-dinero
- https://www.bloomberg.com/news/newsletters/2023-06-20/ai-turing-test-for-chatgpt-or-bard-proposed-by-mustafa-suleyman

Motivación y evolución

Inicialmente, el PLN se desarrolló con el objetivo de reducir la brecha comunicacional entre máquinas y humanos, permitiendo de este modo una interacción más cómoda entre las personas y la tecnología. Casos de uso implementados de hace ya años son, por ejemplo, las interfaces de voz con los ordenadores de abordo en turismos ("pon música", "enciende el aire acondicionado"), los filtros de correo electrónico o los motores de búsqueda.

Hoy en día, sin embargo, se trabaja en optimizar esta comunicación hombre-máquina, habiéndose alcanzado ya niveles que, cuando todo esto comenzó, en 1950, no se hubiera sospechado… ubiquemos, de forma rápida, en la línea temporal, los avances más significativos.

Línea temporal del PLN

1949. IBM patrocina Index Thomisticus, una compilación de escritos de Santo Tomás de Aquino creada por Roberto Busa

1950. Computing Machinery and Intelligence. A.M. Touring.. Test de Touring.

1954. Experimento de Georgetown e IBM traduce del ruso al inglés. Surge la lingüistica computacional.

El planteamiento de este proyecto no fué acertado y tuvo gran influencia en la generación del invierno de la IA.

1956. Conferencia de Darmouth.

Se acuña el término Inteligencia Artificial.

Años 60. Se generan los primeros algoritmos de PLN..

MIT. Terry winograd crea SHRDLU.

Carl Rogers y Joseph Weizenbaum desarrollan ELIZA.

Años 80. Se crean los primeros algoritmos de Aprendizaje Automático. Comienza la generación automática de texto.

La mayoría de los sistemas se basaban en reglas escritas "a mano", pero a finales de esta década se aplica Machine Learning al PLN.

Década 2000. Surge el Deep Learning. .

Se aplican en PLN los modelos ocultos de Markov y, cada vez más, se toman decisiones estadísticas. (principalmente inferencia estadística), que permiten aprender de forma automática, reglas a través del análisis de grandes cantidades de ejemplos del mundo real (Aprendizaje Automático aplicado, definitivamente, al PLN.

Década 2010. Primeros éxitos en la traducción con redes neuronales.

Bengio propone la aplicación de redes Feed Forwardgenerando así el primer "modelo del lenguaje". Losmodelos basados en redes profundas se imponen como nuevo paradigma

Década 2020. Transformers, LLM's y mucho más…

Redes Neuronales destacadas en P.L.N.

Comencemos nuestro recorrido en detalle con las primeras redes neuronales que, realmente, supusieron un hito en la historia del procesamiento del lenguaje natural (y que, todavía hoy, se utilizan como parte de arquitecturas más complejas y avanzadas).

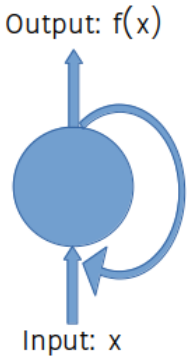
Éstas redes vamos a estudiarlas dada su popularidad histórica, y porque se utilizan actualmente, no solo en algunos casos de forma individual, sino también como parte de modelos avanzados (como componentes en el diseño de arquitecturas complejas), existiendo además modelos prediseñados en librerías tales como "keras", de modo que muy probablemente el lector no va a implementar una red de este tipo desde cero en ningún momento. De cualquier modo, en efecto, es importante conocer su funcionamiento teórico.

R.N.N. Redes Neuronales Recurrentes

Creadas a nivel teórico en la década de los 80, durante muchos años presentaron, debido a limitaciones hardware, serios problemas de entrenamiento.

Sin un diseño de red neuronal pensado para el tratamiento de series de datos grandes y secuenciales, como puede ser el caso de series temporales o, en el caso que nos ocupa, lenguaje natural textos.

Se les denomina "recurrentes" debido a que el la salida retroalimenta a la entrada. Si lo quisiéramos representar gráficamente con una red de una única neurona, el resultado sería muy parecido a este:



Por lo tanto, aunque hasta ahora hemos pensado en neuronas y capas cuya función de activación actúa únicamente hacia adelante, las redes recurrentes vienen a cambiar el estado del arte y permitir que la función de activación retroalimente a la neurona, influyendo en el ajuste de los pesos de la neurona en cada uno de los "time steps" de su entrenamiento.

Como la salida de una neurona recurrente en el instante te tiempo "t+1" se basa en la salida de la misma en el instante de tiempo "t", se dice que, "de algún modo", estas neuronas tienen memoria, "memory cell" o "cell".

Debe comprenderse, llegados a este punto, que esta retroalimentación no es lo mismo que el ya conocido Backpropagation, sino que es un Backprogation entre instantes temporales de entrenamiento, o, como se le denomina de forma habitual, "Backpropagation Through Time".

Exploding graditents y Vanishing gradients.

Un gradiente es una derivada parcial respecto a una de las entradas de la función, y puede representarse como una pendiente, por lo que en nuestro ámbito (aprendizaje profundo), indicará el cambio a realizar en los pesos de una capa de la redc con respecto al cambio en el error.

Teniendo esto en cuenta, se puede hablar de "exploding gradients" -explosivos- cuando el algoritmo asigna una importancia exageradamente alta a los pesos y esto causa un problema de entrenamiento, mientras que podemos referirnos a "vanishing gradients" -desaparecidos- si ocurre lo contrario en la importancia de los pesos, de modo que con "una pendiente" muy próxima a cero, nuestro algoritmo deja de aprender.

Este problema, el de los gradientes desaparecidos, fue el que más tardó en resolverse, y a nosotros nos conduce a presentar las redes LSTM, ya que resuelven el problema del gradiente evanescente, ya que permiten a las RNN recordar sus entradas durante un largo período de tiempo. Esto se debe a que LSTM contiene su información en la memoria, que puede considerarse similar a la memoria de un ordenador, en el sentido que una neurona de una LSTM puede leer, escribir y borrar información de su memoria. Además, las redes LSTM mantienen los gradientes lo suficientemente empinados y, por lo tanto, el entrenamiento es relativamente corto (al tener un delta elevado en el incremento de los pesos, convergen antes) y la precisión de estas redes alta.

Cálculo del estado de memoria "a" en el instante "t".

Con el objetivo de incidir un poco más en la precisión matemática de las neuronas de estas redes, cabe destacar que, normalmente su función de activación es una tangente hiperbólica, terminando en la salida con una función softmax. Siendo, en cada time step, ésta la forma en que se aplica la activación tanh.

Normalmente, una neurona artificial recurrente responde a la siguiente fórmula de activación para el cálculo de su valor de memoria (el que alimenta al time step "t+1")

$$at = \tanh(w_{aa} * a_{t-1} + w_{ax} * x_t + b_a)$$

Debemos considerar que la salida final responderá a la fórmula siguiente:

$$yt = f(w_{ya} * a_t + b_y)$$

Por lo que la salida depende de at que a su vez depende de a_{t-1} y así sucesivamente.

La memoria cortoplacista de las RNN

Vemos en consecuencia que, siendo "a" el valor almacenado en cada instante "t", y que se considera en cada neurona, sin entrar en funciones de activación concretas, el lector puede observar el siguiente efecto instante a instante.

	t_0	t_1	t_2
Activación neuronal:	$f(w_0 * a_0)$	$f(w_1 * a_1)$	$f(w_2 * a_2)$
Desarrollo activación neur.:		$f(w_1 * f(w_0 * a_0))$	
Desarrollo adicional			$f(w_2 * f(w_1 * f(w_0 * a_0)))$

Del cual se deduce que el valor de a0 cada vez es menos significativo, por lo que esta representación matemática explica claramente la memoria cortoplacista de las RNN, ya que cada vez se tiene menos en cuenta los instantes de tiempo anteriores.

Redes LSTM o Long Short Term Memory

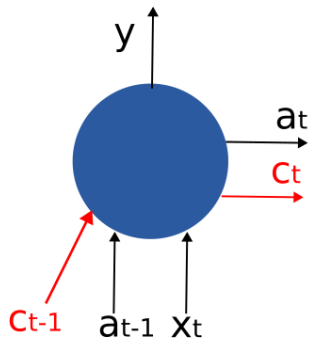
Esta es la arquitectura de redes recurrentes más popular, y se pueden considerar una especialización de las RNN que amplían la memoria de las recurrentes, recordando experiencias importantes (o relevantes) ocurridas y preservándolas por varios instantes de tiempo. Dicho de otro modo, en una red recurrente básica, la memoria simplemente sirve para almacenar la salida de una neurona en un instante t y poderla considerar como parte de la entrada en el instante t+1, es decir, en una red recurrente básica la memoria es a muy corto plazo, mientras que en una red LSTM la memoria es a muy plazo.

Así, una neurona de una LSTM tiene una memoria asociada, y la misma neurona puede leer y escribir en su memoria. Así, esta memoria puede entenderse como una celda, y la neurona decidirá si almacenar información o eliminarla abriendo la puerta de la memoria. La decisión de si almacenar información o no, irá siempre en función de los pesos.

Y es que el problema de una RNN reside en que, al tener una memoria a corto plazo, no permite almacenar estados anteriores significativos, por lo que, en generación automática de texto, por ejemplo, no son capaces de tener en cuenta palabras anteriores, por ejemplo, corriendo el riesgo de que el discurso carezca de sentido o sea erróneo.

Así, una red LSTM, sin embargo, es capaz de recordar, como hemos explicado, estados anteriores relevantes y mantenerlos en memoria por varios instantes de tiempo. Puede intuir el lector, por lo tanto, que se podrán generar textos con más sentido.

Al representar una neurona LSTM, consideramos, con respecto a una neurona recurrente, una entrada adicional y una salida adicional (las salidas alimentarán, igualmente, a la neurona en el instante siguiente:



Neurona LSTM en time step "t"

A estos nuevos datos se les denomina celdas o "memory cells". Debe entenderse la celda de memoria como una cinta de almacenamiento a la que pueden incorporarse datos o eliminar datos que ya no interesan. Los datos almacenados en las "memory cells" se considerarán en la generación de la salida en cada instante t.

Así, se trabaja con puertas, "gates", que permitirán gestionar la información de las celdas:

- forget: Para eliminar datos
- update: Para insertar nuevos datos
- output: Para calcular el valor del estado oculto at teniendo en cuenta las celdas.

Estas "gates" son redes neuronales en sí mismas, que gracias a la función de activación sigmoidea se comportan como válvulas (abierto/cerrado) para permitir la operación (forget/update/output) que corresponda en cada caso.

Nota: El lector puede observar cómo esta función permite una transición muy rápida entre estados binarios, por lo que resulta idónea para la simulación de la apertura o cierre de una compuerta.

Redes Siamesas

Otro avance, sin duda interesantísimo, en estos últimos años es el de las redes siamesas o "Siamese Networks". Este tipo de redes, de naturaleza convolucional (aplican que implementan operaciones matriciales sobre la información, generalmente para extracción de patrones y características en imágenes) se han aplicado, por regla general, para evitar la suplantación de identidad o "identity spoofing", o, si lo prefiere el lector, para la verificación facial. De igual modo, se pude utilizar en la comparación de huellas digitales, por ejemplo, o cualquier otro proceso de identificación biométrica a partir de imágenes.

No obstante, pueden emplearse para la comparación de imágenes con gran parecido, permitiendo alcanzar un nivel de precisión muy elevado en su diagnóstico… existiendo diferentes estudios de interés en este sentido, tales como los siguientes:

- TFG "Redes Convolucionales Siamesas para la Verificación Facial": https://e-archivo.uc3m.es/bitstream/handle/10016/32814/TFG_Asier_Alcaide_Martinez.pdf?sequence=1
- TFG "Utilización de Técnicas de Machine Learning para mejorar la detección de infracción de marcas": https://repositorio.comillas.edu/xmlui/bitstream/handle/11531/46947/TFG%20-%20Armada%20Carrion%2C%20Belen%202.pdf?sequence=2

Lo interesante ocurre, por regla general, cuando se sale de la zona de comfort, lo que en este caso podría empezar por plantearse la siguiente pregunta:

¿Qué aplicaciones de las Redes Siamesas podría haber en NLP?
Bien, a estas alturas, el lector entenderá que las redes siamesas son un buen enfoque para la resolución de problemas de comparación (ya que se componen de dos redes paralelas que comparten pesos entre sí:

Esquema conceptual de una red convolucional siamesa, que explica cómo su output es el nivel de diferencia entre dos imágenes.

Bien, pues respondiendo a la pregunta anterior, probablemente no resultaría descabellado pensar en arquitecturas de red siamesa para la detección de diferencias entre textos (detección de plagios y casos similares).

Debemos comprender que, en las arquitecturas de las convolucionales siamesas, la función de valor busca determinar la distancia que existe entre las características obtenidas, a partir de las dos entradas a comparar. Bien, pues si a partir de textos pudiéramos clasificar cada palabra por un vector de características

numéricas que permitiesen describir cada palabra, podríamos emplear técnicas matriciales para comparar los textos. Este es un tema sobre el que profundizaremos más adelante, los "Word Embeddings".

Una investigación en este sentido de los años 2018 y 2019 propone el uso de arquitecturas Transformer, basadas en el modelo BERT, para extraer los vectores descriptores (embeddings) y poderlos comparar. Se trata del paper titulado "Sentence Embeddings using Siamese BERT-Networks", disponible en https://arxiv.org/pdf/1908.10084.pdf y en la recopilación de materiales esenciales de la asignatura.

Ejercicios prácticos

En este capítulo, que aparece en cada uno de los temas de la asignatura, encontrarás ejercicios resueltos (cuando implican código se tratará explicación y enlace a un cuaderno en gitHub con el ejercicio, para que puedas ejecutarlo en tu entorno de google collaboratory) así como propuestas de ejercicios, de tal forma que, resolviéndolos, puedas afianzar tus conocimientos.

Ejercicio resuelto.

1. Generación de nombres de dinosaurios.

Puedes encontrar el código explicado en https://github.com/jmolina010/RNN_Dinosaurus

Ejercicios propuestos.

1. Copia el ejemplo anterior y sustituye el dataset para emplear una lista de nombres de persona. Ejecuta el código y sonríe al ver el resultado.

2. Implementa una red recurrente que, basándose en el Lazarillo de Tormes, genere texto en castellano antiguo.
Pautas recomendadas:

- Puedes descargar el libro de la web del proyecto Gutemberg. Descárgalo en formato texto. https://www.gutenberg.org/ebooks/320
- En primer lugar, parter del ejercicio 1, generando el texto carácter a carácter. ¿Qué observas en el texto generado? ¿Son palabras con sentido? ¿Y el texto tiene sentido a nivel de oración?
- Si observas problemas o algo incorrecto en el resultado, y teniendo en cuenta la teoría...¿a qué crees que puede deberse?
- ¿Se generan signos de puntuación en lugares inadecuados? Piensa cómo podrías evitar que se generasen signos de puntuación… se llama "limpieza del texto". Forma parte de las tareas fundamentales cuando se prepara un texto para entrenar un modelo.

3. Como continuación del ejercicio anterior, pensemos en lo siguiente:

- Lo que hemos hecho es "enseñar" a nuestro modelo recurrente letras y le hemos entrenado para que, más o menos, las sepa colocar en un orden concreto, por lo que, sinceramente, y dada la escasísima memoria de un modelo recurrente, es posible que nos estemos centrando mucho en las ramas y esto no le permita al modelo ver el bosque para generar un texto que, poco mas o menos, se parezca al del lazarillo.

Hagamos pues, la siguiente prueba….
Nota: Es posible que en tu ordenador o en collab te falte memoria -a no ser que tengas contratado "collab pro". De ser así, reduce el número de itraciones, por ejemplo, de 10000 a 7000 y prueba de nuevo.

- Tokeniza por palabras, generando un diccionario que asigne a cada palabra (no a cada letra) un número. Si trabajas con un corpus (dataset de entrada con el que tokenizar y generar el diccionario) lo suficientemente grande, la red dispondrá de un vocabulario bastante amplio. Emplea, por ejemplo, los tres primeros capítulos del libro. Pista: Piensa en cómo eliminar los signos de puntuación.
- Puede que observes una tasa de error mayor que en los ejercicios anteriores. ¿A qué crees que puede deberse?
  Piensa que se trata de analizar las relaciones entre palabras y determinar qué palabra puede seguir, probablemente, a la anterior. ¿Existe más variedad de combinaciones de palabras al formar una frase o de letras al formar una palabra?
- Si ves que el error es muy elevado y quieres reducirlo, prueba a generar más iteraciones o modificar la estructura de la red. Pista: Debido a un error en keras/tf no se recomienda incrementar demasiado el número de celdas recurrentes (unidades en la capa oculta), ya que podría detenerse el entrenamiento debido al error mencionado, que en el momento de elaborar este ejercicio no se había resuelto (2023/06). No le des importancia a este bug, no afecta a nuestro objetivo ni a las técnicas actuales de PLN.

Para reflexionar al resolver el ejercicio:

- El texto generado, ¿emplea palabras del castellano antiguo?
- ¿Tiene sentido el texto generado?
- Utiliza el mismo modelo, pero aplica un texto de entrenamiento de una noticia de actualidad que sea lo suficientemente extensa y observa los cambios en la salida. ¿Podrían emplearse este tipo de redes neuronales para generar artículos de un área de conocimiento específica? ¿Por qué?

En el siguiente tema abordaremos las word embeddings, una técnica avanzada para tokenizar palabras manteniendo su significado y, a partir de esta técnica, haremos ejercicios con redes LSTM.

Ahora, y antes de abordar el siguiente tema, te recomiendo que amplíes conocimientos a partir de la siguiente selección de referencias. Lectura suave, no estudio, te servirá para abordar con más interés y facilidad los siguientes capítulos.

Referencias

En esta sección se propone un conjunto de referencias orientadas a ampliar los conocimientos adquiridos durante el presente bloque de contenidos.

1. "Intelligent Machinery". A.M. Turing, 1948
2. "Computing Machinery and Intelligence". A.M. Turing, 1950
3. "Redes Neuronales Convolucionales Siamesas Verificación Facial". TFG 2019 – 2020, Asier Alcaide Martínez
4. "Utilización de Técnicas de Machine Learning para mejorar la detección de infracción de marcas", TFG 2021, Belén Armada Carrión
5. "Sentence Embeddings using Siamese BERT-Networks", Nils Reimers and Iryna Gurevych, 2019
6. "Python Deep Learning". Ed. Marcombo, Aut. Jordi Torres. Libro muy interesante para iniciarse en Deep Learning. Incluye un capítulo que aborda las redes neuronales Recurrentes.