Anales de Lingüística. Segunda época
n.7 – AGO-DIC 2021. Mendoza, Argentina ISSN 0325-3597 (impreso) - ISSN 2684-0669 (digital)
https://revistas.uncu.edu.ar/ojs3/index.php/analeslinguistica
pp. 145-174

# Modelos de aprendizaje profundo para comprensión de textos y una implementación prototípica de GPT-2 para una tarea específica de generación de lenguaje natural

Deep Learning-based Natural Language Understanding Models and a Prototype GPT-2 Deployment Fine-Tuned for a Specific Natural Language Generation Task

**Fernando Balbachan**[1] https://orcid.org/0000-0003-0823-180X
fernando.balbachan@natural.do
Universidad de Buenos Aires, Argentina, Universidade Tuiuti do Paraná, Brasil
y Natural Tech, Argentina

**Natalia Flechas**
nataliaflechas@outlook.com
Universidad de Buenos Aires, Argentina

**Ignacio Maltagliatti** https://orcid.org/0000-0002-6644-4750
ignacio.maltagliatti@natural.do
Universidad Tecnológica Nacional, Argentina
y Natural Tech, Argentina

**Francisco Pensa**
francisco.pensa@natural.do
Natural Tech, Argentina

**Lucas Ramírez**
lucas.ramirez@natural.do
Natural Tech, Argentina

[1] Recibido: 24.02.2021 | Aceptado: 21.04.2021

**Resumen**

Desde el año 2013, el paradigma conexionista en procesamiento de lenguaje natural (PLN) ha venido resurgiendo en ámbitos académicos a partir de nuevas arquitecturas para luego ser adoptado en la industria de software. Este paradigma hace uso de poderosos recursos de cómputo, en una revolución algorítmica conocida como aprendizaje profundo (*Deep Learning*). Numerosas y sucesivas propuestas superadoras se han ofrecido en una vertiginosa carrera por obtener métricas (*benchmarking*) que se acercaran al estado del arte para tareas generales de PNL, según diversos estándares (BLEU, GLUE, SuperGLUE). A partir de 2018, con la revolución de los *transformers* en los últimos dos años (ELMo, BERT y GPT-2), los modelos de Deep Leaning atrajeron aún más el interés de la comunidad científica, de la industria y de neófitos. En este artículo, proponemos una sucinta pero exhaustiva historización de los modelos que han venido evolucionando durante esta revolucionaria última década y ofrecemos, a modo de ejemplo ilustrativo, una arquitectura de implementación completa de Deep Learning para el modelo de código abierto más reciente GPT-2, entrenado para una tarea específica de generación de slogans comerciales en cualquier segmento de producto.

**Palabras clave**: aprendizaje profundo, ELMo, BERT, GPT-2, comprensión del lenguaje natural, generación de texto.

**Abstract**

Since 2013, the connectionist paradigm in Natural Language Processing (NLP) has resurged in academic circles by means of new architectures to be adopted later by the software industry with the use of great computing power. It is a truly algorithmic revolution, known as Deep Learning. Several models have been offered in a speedy race in order to improve state-of-the-art metrics for general domain NLP tasks, according to the most frequentlly used standards (BLEU, GLUE, SuperGLUE). From 2018 onwards, Deep Learning models have attracted even more attention through the so-called *Transformers* revolution (ELMo, BERT y GPT-2). In this paper, we propose a brief yet exhaustive survey on the models that have been evolving during this last decade. We also describe in detail a complete from scratch implementation for the most recent open-source model GPT-2, fine-tuned for a specific NLG task of slogan generation for commercial products.

**Keywords**: Deep Learning, ELMo, BERT, GPT-2, Natural Language Understanding (NLU), Natural Language Generation (NLG).

## Introducción

En las últimas décadas, la investigación en redes neuronales ha experimentado diversos giros. Son tres las olas de investigación que el paradigma conexionista —como también se lo conoce— ha visto hasta la fecha. La primera data de finales de la década de los cincuenta; la segunda, durante los años ochenta, y la última, posibilitada por las capacidades del nuevo hardware, comienza en la primera década de este siglo y aún continúa vigente. Este hecho implicó, en el área de lingüística computacional y procesamiento del lenguaje natural (PNL), el uso cada vez más prevalente de las redes neuronales en los sistemas de comprensión y producción de lenguaje, en detrimento de los enfoques de corte simbólico utilizados a finales del siglo XX y del más reciente paradigma estadístico (*machine learning*), dominante durante la primera década del nuevo milenio.

A la luz de los avances recientes en el campo, los objetivos de este trabajo son: 1) brindar una sistematización general de los avances en modelos de entendimiento de lenguaje (*Natural Language Understanding* NLU) en los últimos años, 2) explicar en detalle algunos de los modelos pre-entrenados más recientes en el resurgido paradigma conexionista, también conocido como arquitectura *Deep Learning*, y 3) presentar un caso práctico de implementación de alguna de estas arquitecturas en una tarea específica de NLP: generación de slogans publicitarios para productos de cualquier dominio en lengua portuguesa.

Con estas metas en mente, en la sección 1, pasamos revista de manera general a la evolución de los algoritmos basados en redes neuronales desde su adopción a principios de la década de 2010. Luego, brindamos explicaciones detalladas de los modelos que no sólo han supuesto la superación del estado del arte en diversas tareas de evaluación, sino que han sido influyentes en las arquitecturas de los modelos que los preceden, comenzando por *Sequence-to-Sequence Learning* (Sutskever et al. 2014), Semi-*supervised Sequence Learning* (Dai y Le 2015) y *Universal Language*

*Model Fine-tuning for Text Classification* (ULMFiT) (Howard y Ruder 2018). También mencionaremos los modelos pre-entrenados como más recientes alrededor de la arquitectura denominada *Transformer*, basado en una pila de codificadores y decodificadores con capas de *Self-Attention* y *Feedforward* (Vaswani et al. 2017); ELMo (Parsers et al. 2018); BERT, un modelo *Transformer* grande (Devlin et al. 2018), y finalmente GPT-2, cuya arquitectura también se basa en la del *Transformer* (Radford et al. 2019).

## 1. Algoritmos basados en Deep Learning para tareas estándares generales de NLP/NLU

Los años 2013 y 2014 marcan el momento de un renacimiento del paradigma conexionista, en el que los modelos neurales fueron adoptados en las tareas de procesamiento de lenguaje natural por la industria del software, habiendo experimentado un verdadero renacimiento en el ámbito académico. De particular importancia fue el uso de las redes neuronales recurrentes, convolucionales y recursivas. A continuación, mencionaremos brevemente la utilización de cada uno de estos modelos en NLP/NLU.

Las Redes Neuronales Recurrentes (*Recurrent Neural Networks* o RNNs) resurgieron en la comunidad académica al ser capaces de lidiar con tareas de clasificación de patrones en donde la entrada (*input*) y la salida (*output*) son secuencias dinámicas. Esto es naturalmente deseable cuando se trabaja con lenguaje natural. Para lograr esta tarea, las RNNs representan una secuencia con un vector (también conocido como *hidden state vector* o vector de estados ocultos) de dimensionalidad fija que incorpora nuevas observaciones usando funciones no lineales (Sutskever et al. 2014). Si bien se pensaba que la adopción de este tipo de redes significaría una gran dificultad a la hora de su entrenamiento, Sutskever (op.cit.), demostró que éste no era necesariamente el caso. En cuanto al subtipo de redes, si bien en un principio se hizo uso de RNNS vainilla[1] (Elman 1990), luego se pasó a

1 Red neuronal básica de una sola capa cuya salida se utiliza como “salida actual” (externa) y al mismo tiempo como “estado actual” de la celda o módulo de la RNN.

las arquitecturas LSTM-RNN (*Long-Short Term Memory*) *(*Hochreiter y Schmidhuber 1997*),* cuya principal virtud es la resistencia a los problemas de desvanecimiento de la gradiente[1] y de gradiente explosivo[2]. Las LSTM bidireccionales, por su parte (Graves et al. 2013), son utilizadas para lidiar con los contextos a izquierda y derecha ya que son capaces en cualquier momento de preservar información tanto del pasado como del futuro. Por ejemplo, si se desea predecir la siguiente palabra en una oración, lo que verá un LSTM unidireccional en su entrenamiento es:

"*Los chicos fueron a* ..."

A partir de este contexto, intentará inferir la siguiente palabra. Una red LSTM bidireccional puede ver información hacia adelante (derecha):

"*Los chicos fueron a* ..."

Y hacia atrás (izquierda):

"... *y luego salieron de la piscina*"

Con la información del futuro podría ser más fácil para la red aprender cuál es la siguiente palabra.

La arquitectura LSTM tiene una estructura en forma de cadena (secuencial) donde el módulo o celda repetida tiene características particulares con cuatro capas que interactúan de una manera especial. En la **figura 1**, en la celda de repetición, cada línea lleva un vector completo, desde la salida de un nodo hasta las entradas de otros. Los círculos de la parte superior representan operaciones puntuales, como la suma de vectores. Los cuadros más pequeños representan capas de redes neuronales aprendidas.

1 Algunas funciones de activación con derivada menor a 1 producen en redes con muchas capas, así como en redes recurrentes un desvanecimiento de la estimación del gradiente en cada iteración ya que se multiplica varias veces por valores pequeños (el gradiente tiende a 0) impidiendo eficazmente actualizar los pesos de las redes.

2 En contraposición al problema de desvanecimiento del gradiente, cuando se usan funciones de activación cuyas derivadas toman valores muy grandes (mayor a 1) el gradiente puede crecer exponencialmente en cada iteración y el problema de optimización podría no aproximarse suficiente al mínimo global.

Las líneas que se fusionan denotan concatenación, mientras que una bifurcación de líneas indica que su contenido se está copiando y las copias van a diferentes ubicaciones.

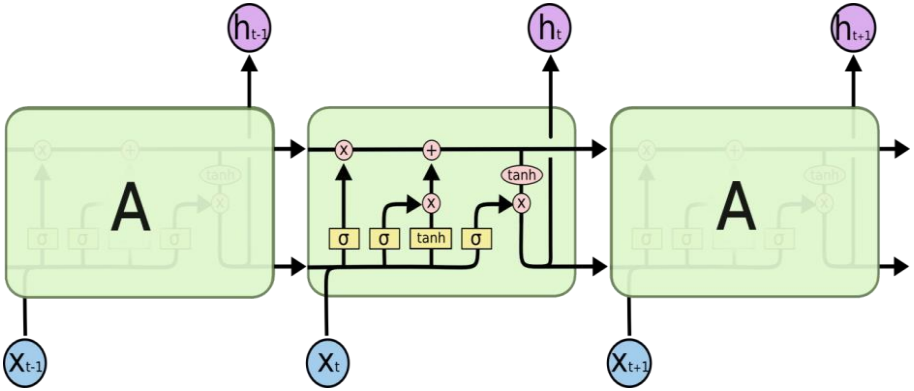


Figura 1: Arquitectura LSTM. El módulo de repetición contiene cuatro capas que interactúan.

Las Redes Neuronales Convolucionales (CNNs, por sus siglas en inglés) tuvieron su auge en las tareas relacionadas con el desarrollo de visión artificial, para luego ser adoptadas en las tareas de NLP (Kalchbrenner et al. 2014; Kim et al. 2014). En este último campo, las CNNs operan en dos dimensiones, con el requerimiento de mover los filtros[1] sólo en la dimensión temporal. Son más paralelizables[2] que las RNNs, ya que su estado en cada paso sólo depende del contexto local y no de todos los estados pasados[3]. Pueden ser extendidas con campos receptivos más

1 Conjunto de kernels o matrices que "recorre" todas las neuronas de entrada de la red (de izquierda-derecha, de arriba-abajo) que se utiliza para generar una nueva matriz de salida o capa de neuronas ocultas, operando matemáticamente mediante producto escalar.

2 Capacidad de usar dos o más CPU's para la ejecución de uno o varios procesos informáticos.

3 Ciertas aplicaciones de clasificación de texto, análisis de sentimientos, etc. no necesitan utilizar la información almacenada en la naturaleza secuencial de los datos. Por ejemplo, en la reseña hipotética de un restaurante "*me decepcionó mucho este restaurante, el servicio fue increíblemente lento, la comida mediocre y no volveré*" si bien hay información secuencial en los datos, para predecir si el sentimiento fue bueno o malo la información relevante se encuentra en las frases "*me decepcionó*", "*increíblemente lento*" y "*mediocre*", o sea, en su contexto local.

amplios, haciendo uso de convoluciones dilatadas[1] para capturar un contexto mayor (Kalchbrenner et al. 2016). Las CNNs y las arquitecturas LSTM pueden ser combinadas y apiladas (Wang et al. 2016).

Dos características fundamentales del lenguaje natural son la organización jerárquica y la recursividad. Al respecto, tanto las redes del tipo RNN como las LSTM tienen precisamente la desventaja de tratar al lenguaje como si fuera una secuencia, en detrimento de las dos características antes mencionadas. Este hecho da paso al uso de las Redes Neurales Recursivas (Socher et al. 2013) donde la secuencia de entrada tiene que procesarse jerárquicamente en forma de árbol, es decir, construyen la representación de la secuencia de manera *bottom-up* (ascendente) y no de derecha a izquierda o izquierda a derecha. En cada nodo del árbol de procesamiento del *input*, una nueva representación es computada mediante la composición de las representaciones de los nodos hijos. Las arquitecturas LSTM también han sido adaptadas para el uso con esta clase de redes, ya que un árbol se puede entender como una imposición de orden de procesamiento distinto sobre una RNN (Tai et al. 2015).

En todo caso, trabajar con estructuras jerárquicas no significa limitarse al uso de LSTMs o RNNs. Otras opciones incluyen el aprendizaje de incrustaciones de palabras (*Word Embeddings*) del contexto gramatical (Levy y Goldberg 2014); la generación por parte de los modelos del lenguaje de palabras basada en una pila sintáctica (Dyer et al. 2016); y el uso de Redes Neuronales de Grafos Convolucionales (*Graph Convolutional Neural Networks*), que pueden operar sobre árboles (Bastings et al. 2017).

En 2014 se publica la propuesta de *Sequence-to-Sequence Learning* (Sutskever et al. 2014), que tiene como objetivo mapear una entrada de longitud fija con una salida también de longitud fija donde el tamaño entre

1 Convolución es el tratamiento de una matriz por otra que se llama “kernel”. Convolución dilatada es una matriz con espacios o “gaps” definidos. Una tasa de dilatación k = 1 es la convolución normal; k > 1 expande el campo de cobertura (contexto) sin pérdida con similares costos de cálculo. La expansión se genera con espacios insertados entre elementos del kernel (se forma una matriz dispersa).

ambos vectores (entrada y salida) puede diferir. Por ejemplo, traducir *"¿Qué estás haciendo hoy?*" del español al chino tiene entrada de 4 palabras y salida de 7 símbolos (今天 你 在 做 什麼?); no se puede usar una red LSTM normal para mapear cada palabra de la oración en este idioma a la oración en chino, por eso se recurre a este tipo de arquitecturas.

La propuesta de Attention (Bahdanau et al. 2015) es hoy en día un elemento clave cuando se habla de redes neuronales. Se puede definir la función de *Attention* como el mapeo de una consulta (*query*) y un conjunto de pares atributo-valor a un output, en donde la consulta, atributos, valores y output son vectores. El output se computa como la suma ponderada de valores, en donde el peso asignado a cada valor es calculado por una función de compatibilidad de la consulta con una clave correspondiente (Vaswani et al. 2017).

*Attention* representa una mejora de la propuesta de Sutskever et al. (2014), ya que el principal problema del aprendizaje *Sequence-to-Sequence* consistía en comprimir el contenido entero de la secuencia fuente en un vector de tamaño fijo. El mecanismo de *Attention* provee una solución al permitir que el decodificador acceda a los estados ocultos de la secuencia de fuente, que después son provistos en forma de media ponderada (Ruder 2018). Este mecanismo es ampliamente utilizado y de potencial utilidad para cualquier tarea que requiera decisiones basadas en ciertas partes del input. Además, no está solamente restringido a consultar el input, sino que también puede ser utilizado para acceder al contexto que rodea dicho input, con lo que se logra una representación de palabras más sensible al contexto. En la propuesta de Vaswani et al. (2017), múltiples capas de *Self-Attention* son la característica principal de la arquitectura del modelo *Transformer*.

En las siguientes secciones, discutiremos en profundidad los modelos de lenguaje más recientes, pues su auge ha despertado gran interés en la comunidad académica. Esto porque desde el 2013, con la aparición de la técnica de *Word Embeddings* o word2vec (Mikolov et al. 2013), los

proyectos de NLP y Deep learning han seguido por lo general la misma receta: el pre-entrenamiento para un NLU global con grandes cantidades de datos mediante algoritmos como el ya mencionado word2vec (Pennington, Socher y Manning 2014) y el ajuste fino (*fine-tuning*) en una tarea específica.

Así pues, los modelos del lenguaje para un NLU global tan sólo requieren texto sin etiquetar, por ello, el entrenamiento puede consistir en millones de tokens, dominios y lenguas completamente nuevas. Al permitir un entrenamiento con datos no etiquetados, los modelos pre-entrenados son particularmente útiles en aquellos contextos en los que datos con etiquetas son escasos, difíciles o costosos de conseguir.

Los siguientes apartados explican con más detalle en qué consisten estos modelos, cómo está configurada su arquitectura, y cómo se posicionaron con respecto al estado del arte en sus respectivos contextos de surgimiento a lo largo de la evolución del paradigma durante la última década. Sin embargo, antes de pasar a las siguientes secciones, queremos dedicar unas palabras a la pregunta de cómo se transfiere la información de un modelo del lenguaje pre-entrenado a una tarea específica (*fine-tuning*).

Por un lado, los *embeddings* del modelo del lenguaje pueden ser utilizados como parámetros en un modelo meta (*target*), en otras palabras, se hace uso del modelo pre-entrenado como un extractor de parámetros y se incorporan las representaciones como parámetros en otro modelo inicializado aleatoriamente. También es posible hacer *fine-tuning* con los datos de una tarea meta a todo el modelo de lenguaje (Ramachandran et al. 2017; Howard y Ruder 2018) (ver sección 2 para un caso práctico de un prototipo de arquitectura GPT-2 aplicada a una tarea creativa específica de *Natural Language Generation* NLG).

### 1.1. *Sequence*-to-*Sequence Learning with Neural Networks* (Sutskever et al. 2014)

El modelo *Sequence-to-Sequence Learning* fue diseñado para convertir una secuencia de tokens en una lengua *a* a otra secuencia de tokens en otra lengua *b*. Si bien las redes neurales profundas (DNNs, por sus siglas en inglés) son un método poderoso, con un desempeño destacado en tareas de reconocimiento de voz e imagen, el modelo propuesto por los autores de este artículo nace en torno a las dificultades de las DNNs para resolver problemas que impliquen manejar inputs y targets no codificados con vectores de dimensiones fijas, pues en el procesamiento de lenguaje natural, es común enfrentarse a tareas que utilizan vectores cuyas dimensiones, a priori, son desconocidas (*data sparsity*). Así, para poder manejar secuencias de estas características, los autores proponen un método independiente de dominio que aprenda a mapear de secuencias a secuencias. La solución consiste en hacer uso de LSTMs: una *Multilayered LSTM* utilizada para mapear la secuencia de input, un paso a la vez, a un vector de dimensionalidad fija. Después, una LSTM decodificadora extrae de este vector una secuencia de output. Esta última LSTM es una RNN en esencia, sólo que condicionada a la secuencia de input. La **figura 2** ilustra este proceso.

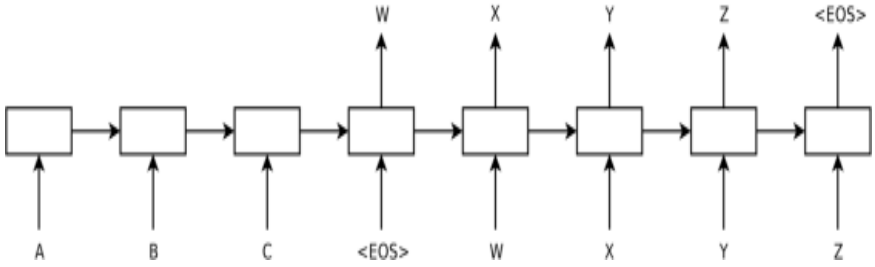


Figura 2: El modelo lee la secuencia de input “ABC” y produce la secuencia “WXYZ” como oración de output. Cuando predice el token de fin de secuencia “end-of-sentence token” ("<eos>") la predicción de la secuencia de salida está completa (no se realizan nuevas inferencias) (Sutskever, Vinyals y Le 2014).

La arquitectura del modelo se divide en 4 capas de 1000 perceptrones, cada una almacenada en una GPU (ordenadores de alto poder de cómputo paralelizable, en comparación con las tradicionales CPUs). El modelo recurre a incrustaciones de palabras de 1000 dimensiones; un vocabulario de input de 160.000 palabras y de output de 80.000. El modelo en total tiene 384 millones de parámetros utilizados en la fase entrenamiento, de los cuales, 64 millones son solo conexiones recurrentes.

El modelo fue entrenado con la tarea de traducción Inglés-Francés WMT ’14, y evaluado utilizando el algoritmo BLEU. El resultado de la evaluación supera el estado de la cuestión (*state-of-the-art* o SOTA) en esa época.

### 1.2. Aprendizaje semi-supervisado de secuencias (*semi*-supervised *Sequence Learning*) (Dai y Le 2015)

El modelo propuesto por Dai y Le (2015) significa darle una vuelta de tuerca al de Sutskever et al. (2014), para lograr un aprendizaje no supervisado que pueda hacer uso de datos sin etiquetar. El método se resume en la utilización, en primera instancia, de un auto-codificador sin supervisión que es pre-entrenado con una LSTM, de donde se obtienen parámetros utilizados, posteriormente, en la inicialización de una LSTM estándar para una tarea de entrenamiento supervisada. Por esta razón, el sistema es llamado SA-LSTM.

En cuanto a la evaluación, la SA-LSTM fue capaz de superar los cinco benchmarks, alcanzando o superando todas las baselines anteriores.

### *1.3. Universal Language Model Fine-tuning for Text Classification* (ULMFiT) (Howard y Ruder 2018)

ULMFiT es un método de aprendizaje de transferencia que puede ser aplicado a cualquier tarea de NLP. ULMFiT se caracteriza por tres etapas: a) el entrenamiento de un modelo de lenguaje con un corpus de dominio general y de grandes dimensiones; b) El *fine-tuning* del modelo del lenguaje para la tarea objetivo. Sobre este último punto, los autores proponen nuevas técnicas de *fine-tuning* (*slanted triangular learning rates* y

*discriminative fine-tuning*) que previenen el olvido catastrófico —definido como la tendencia de una red neuronal de olvidar completa y abruptamente la información previamente aprendida después de aprender información nueva— y facilitan el aprendizaje robusto en diversas tareas; c) El *fine-tuning* del clasificador, utilizando las técnicas de *gradual unfreezing* y *discriminative fine-tuning*. La **figura 3** ilustra estos pasos.

El modelo es universal ya que funciona en diversas tareas cuyo número y tamaño de documentos y etiquetas varían. Además, usa una sola arquitectura y proceso de entrenamiento, no requiere ingeniería personalizada de rasgos (*custom feature engineering*) o preprocesamiento y tampoco documentos o etiquetas adicionales en el dominio.

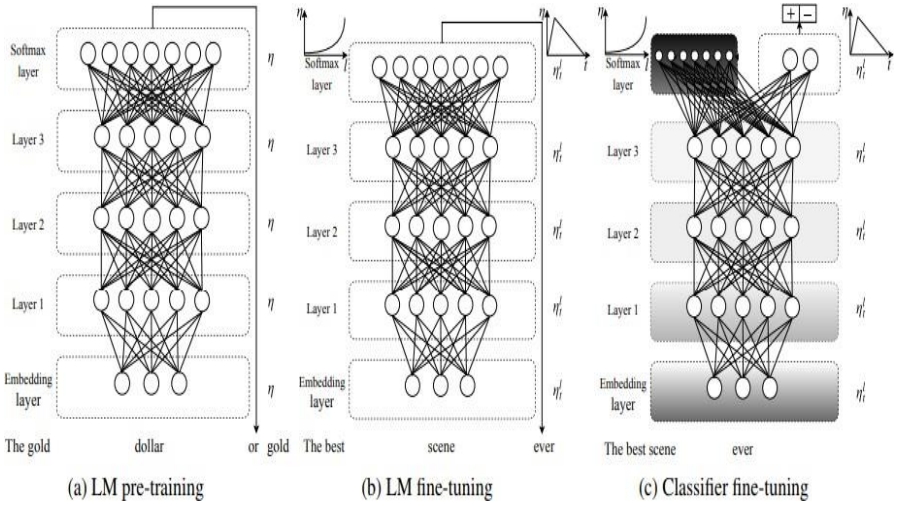


Figura 3: Pasos característicos de ULMFIT: a) (pre)-entrenamiento, b) ajuste a tarea target (fine-tuning) y c) inferencia o clasificación.

En cuanto a sus características, los autores hacen uso del modelo AWD-LSTM (Merity et al. 2017), que consiste en una arquitectura LSTM regular sin adicionales como *attention*, *short-cut connections*, etc. En la primera etapa, vale mencionar que el modelo es entrenado con el corpus Wikitext-

103, que consiste en 29.595 artículos de Wikipedia preprocesados y 10 millones de palabras. El método supera el estado del arte en seis tareas de clasificación de textos.

### *1.4.* La revolución de los *Transformers*

El *Transformer* es un modelo revolucionario, que se propone como una alternativa a los modelos de transducción de secuencias basados en RNN o. El Transformer se caracteriza por la simplicidad de su arquitectura de red basada sólo en mecanismos de *Attention*, sin recurrencias o convoluciones (**figura 4**). Si bien en el artículo original el modelo es testeado en tareas de traducción automática, la relevancia del modelo para PLN se hace palpable en la influencia que la arquitectura del *Transformer* tendrá en los modelos pre-entrenados que lo suceden, como veremos más adelante.

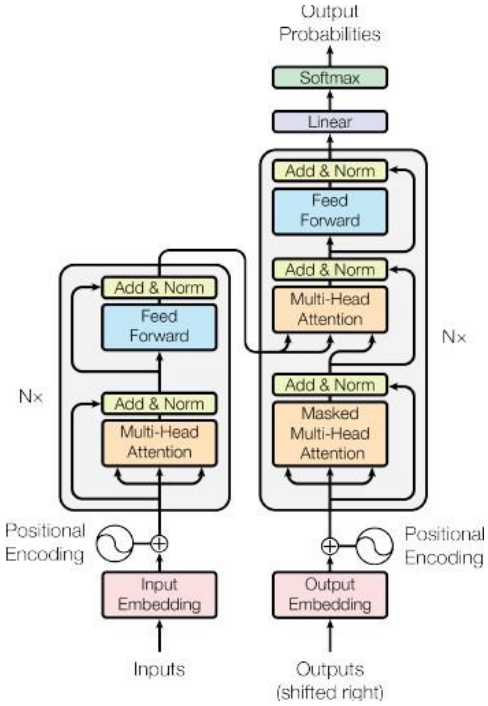


Figura 4: Arquitectura del Transformer (Vaswani et al. 2017)

En cuanto a su arquitectura, el *Transformer* cuenta con un componente de codificación y otro de decodificación. El componente de codificación consiste en una pila de seis codificadores. Todos los codificadores tienen una estructura idéntica, cada uno cuenta con dos capas, la primera es un mecanismo de *Self-Attention* con múltiples núcleos (*multi-head self-attention mechanism*), a través del cual se procesa el input. El output de esta primera capa pasa luego por la segunda, una red *Feed-forward*. Se implementa, además, una conexión residual alrededor de cada subcapa, seguida por la normalización de la capa. Para facilitar las conexiones residuales, todas las subcapas del modelo, además de las capas de embeddings producen outputs de dimensión $d_{model} = 512$. Por último, la subcapa de *Multi-head Attention* se modifica mediante el enmascaramiento, que previene que para la predicción de la posición *i* se preste atención a posiciones subsecuentes.

Anteriormente habíamos introducido el concepto de *Attention stand alone* o *Self-Attention* Para retomar el concepto de *Self-Attention* en esta nueva versión mejorada, quizá sirva pensar en un pronombre anafórico, tal como *ella* en la oración "*la gallina cruzó la calle, porque ella estaba muy cansada*". Para un humano, encontrar el referente del pronombre parece una tarea trivial. Sin embargo, para un algoritmo no lo es. Aquí entra en juego *Self-Attention*: a medida que el modelo procesa cada palabra, *Self-Attention* permite que mire a otras posiciones en el input en busca de pistas que lo puedan ayudar a codificar la palabra. En otras palabras, *Self-Attention* es el método mediante el cual el modelo integra la comprensión de otras palabras relevantes a la que estamos procesando actualmente. Pero en la arquitectura del *Transformer*, además, se refina la capa de *Self-Attention* implementando el uso de *Multi-Head Attention*, lo cual expande la habilidad del modelo de concentrarse en posiciones distintas. Como última característica, nos gustaría destacar el uso de *Positional Encodings*, recurso mediante el cual se inyecta información sobre la posición relativa o absoluta de los tokens de una secuencia, sin hacer uso de recurrencias o convoluciones. Esta información es añadida a los *embeddings* de input en

el fondo de las pilas de codificadores y decodificadores y tiene la misma dimensión que los *embeddings*, de manera tal que ambos vectores puedan ser sumados.

Para el entrenamiento del modelo se hizo uso de los datasets WMT 2014 English-German (4.5 millones de pares de oraciones) y WMT 2014 English-French (36 millones de oraciones). En los resultados de la evaluación, el modelo logra mejorar los resultados existentes en BLEU. Además, el *Transformer* generaliza bien en otras tareas, como mostró la prueba de parsing de constituyentes con datos en cantidades tanto masivas como limitadas.

### 1.5. ELMo (Peters et al. 2018)

Los autores de ELMo (*Embeddings from Language Models*) se propusieron introducir una nueva clase de representación de palabras, caracterizadas por su contextualización profunda que logra capturar características del uso de palabras relacionadas con su sintaxis y semántica, y cómo estos usos varían en distintos contextos lingüísticos (en otras palabras, es capaz de dar cuenta de la polisemia), en contraste con otras representaciones pre-entrenadas de palabras (Mikolov et al. 2013; Pennington et al. 2014).

Existen diferencias con los enfoques que pretenden aprender vectores de palabras contextualizados: las representaciones ELMo son profundas, ya que se computan encima de bi-LMs (*bidirectional Language Models*) de dos capas con convoluciones de caracteres, como una función lineal de los estados internos de la red neuronal. Esto permite llevar a cabo aprendizaje semi-supervisado, que consiste en el pre-entrenamiento del bi-LM a gran escala, con un corpus masivo de aproximadamente 30 millones de oraciones. Las representaciones ELMo pueden ser fácilmente integradas a modelos existentes, lo que conlleva a un puntaje más elevado varias tareas que sobrepasa el estado del arte.

El modelo final utiliza dos capas LSTM con 4096 unidades y 512 dimensiones de proyección y una conexión residual desde la primera hasta

la última capa. Así, dado un bi-LM pre-entrenada y una arquitectura supervisada para una tarea de NLP meta, usar lo primero para mejorar los resultados de lo segundo es sencillo: hay que hacer correr el biLM y registrar todas las representaciones de cada capa, para cada palabra, luego, se deja que el modelo de la tarea específica aprenda una combinación lineal de dichas representaciones.

Mediante ablaciones y otro tipo de experimentos, los autores comprueban que cada capa del bi-LM codifica diferentes tipos de información sintáctica y semántica sobre las palabras en contexto, y que usar todas las capas mejora el desempeño. También se ve, por una parte, que las mejoras con ELMo son más notorias para sets de datos más pequeños, y por otra, que la cantidad de datos de entrenamiento necesarios para alcanzar un nivel aceptable de desempeño se ve reducido**.**

### 1.6. BERT (Devlin et al. 2018)

BERT, llamado así por su nombre extendido en Inglés (*Bidirectional Encoder Representations from Transformers*), está diseñado para pre-entrenar representaciones bidireccionales profundas mediante el condicionamiento conjunto tanto del contexto izquierdo como el derecho en todas las capas. Las representaciones pre-entrenadas BERT pueden ser refinadas (*fine-tuning*) con tan solo una capa adicional de output para crear modelos multi-dominio que superan el estado del arte, sin modificaciones sustanciales de las arquitecturas específicas a una tarea. Su punto de partida es la crítica hacia las técnicas utilizadas hasta ese momento, que limitaban el poder de las representaciones pre-entrenadas, problema particularmente saliente en los enfoques que incluyen fine-tuning, aunque también en aquellos enfoques basados en rasgos. La limitación principal está en el hecho de que la mayoría de modelos del lenguaje son unidireccionales, lo que limita las opciones de arquitectura que pueden ser utilizadas en el pre-entrenamiento. Esto contrasta, por un lado, con el modelo GPT de OpenAI (Radford et al. 2018), que se caracteriza por una arquitectura de izquierda a derecha, en donde cada token solo puede

prestar atención a tokens previos en las capas de *self-attention* del *Transformer* y, por otro lado, con ELMo, que concatena de manera superficial LMs de izquierda-derecha y derecha-izquierda independientemente entrenados.

Una de las novedades que introducen los autores para mejorar los enfoques basados en *fine-tuning* es proponer un nuevo objetivo de pre-entrenamiento: el modelo de lenguaje enmascarado o MLM (*masked language model*). El MLM enmascara de manera aleatoria algunos de los tokens del input, y el objetivo consiste en predecir el ID de vocabulario original de la palabra enmascarada. Así, a diferencia del entrenamiento de izquierda a derecha de un modelo del lenguaje, el nuevo objetivo permite que la representación fusione los contextos a la izquierda y derecha, lo que a su vez permite entrenar un *Transformer* bidireccional profundo. También se introduce una tarea de predicción de la siguiente oración.

Además de mostrar la importancia del pre-entrenamiento bidireccional para las representaciones del lenguaje como la contribución más importante del modelo, los autores también dan prueba de que las representaciones pre-entrenadas eliminan la necesidad de arquitecturas altamente elaboradas específicas a una tarea particular.

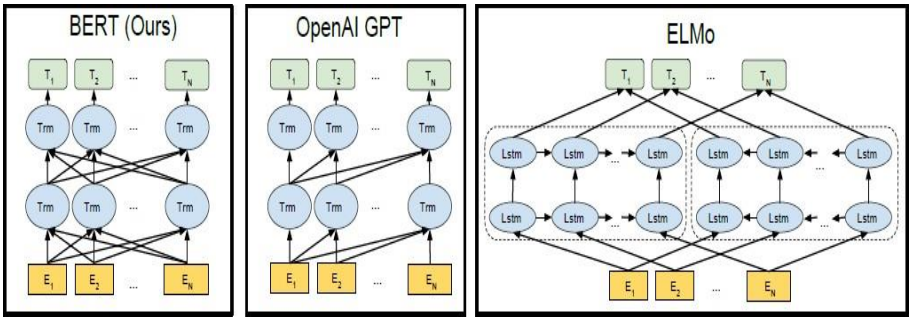


Figura 5: Comparación de distintas arquitecturas de modelos de pre-entrenamiento. BERT usa un *Transformer* bidireccional. El GPT de OpenAi, un *Transformer* de izquierda a derecha y ELMo, concatena LSTMs de izquierda a derecha y de derecha izquierda independientes que generan parámetros para tareas meta.

BERT supera el estado del arte en 11 tareas de PLN de sets de evaluación como GLUE, SQuAD, SWAG y la tarea de reconocimiento de entidades CoNLL-2003. La **figura 5** ilustra esta comparación.

### 1.7. *Generative Pre-trained Transformer* o GPT-2 (Radford et al. 2019)

En *Language Models are Unsupervised Multitask Learners* (Radford et al. 2019), los autores pretenden demostrar que varias de las tareas de NLP (traducción automática, resumen, comprensión lectora, etc.) son aprendidas por modelos del lenguaje sin ninguna supervisión explícita, al ser entrenados en un nuevo set de datos de millones de páginas web llamado WebText, un corpus de 40GB obtenido mediante *web crawling*, caracterizado por la representación variada de dominios y contextos que demuestren varias de las tareas estándar de PLN. Los autores parten de la observación de que los sistemas entrenados mediante grandes sets de datos, modelos de alta capacidad y aprendizaje supervisado son frágiles y sensibles a cambios leves en la distribución de los datos y la especificación de las tareas. Para estos autores, los sistemas actuales son mejor caracterizados como expertos limitados que como generalistas competentes (Radford et al. 2019), lo cual lleva a la necesidad de diseñar sistemas generalizados que puedan desempeñarse en una plétora de tareas sin la necesidad de sets de datos meticulosamente etiquetados según el tipo de tarea.

Una de las causas de la falta de generalización es la prevalencia del entrenamiento en una única tarea en vez de una miríada amplia tanto de tareas como de dominios. Los mejores modelos que usan una combinación específica de pre-entrenamiento y refinamiento (*fine-tuning*) requieren de supervisión significativa.

Así, GPT-2 pretende demostrar que los modelos de lenguaje pueden realizar tareas meta en un contexto de insuficiencia de datos de entrenamiento para algunas de las clases (*zero-shot setting*), sin ninguna modificación de parámetros o arquitectura. Se basa en la especulación de que un modelo del lenguaje con una capacidad suficiente aprenderá a

inferir y resolver tareas mostradas en secuencias de lenguaje natural para poder predecirlas mejor. A esto llaman aprendizaje multitarea no supervisado.

Los modelos del lenguaje se basan en la ya clásica arquitectura del *Transformer*, siguiendo los detalles del modelo de OpenAI, GPT (Radford et al. 2019) con ciertos cambios: a diferencia de BERT, la segunda versión de GPT (GPT-2) usa bloques decodificadores de *Transformer*, mientras que BERT utiliza bloques codificadores.

Otro rasgo a destacar consiste en que cada vez que GPT-2 produce un token, dicho token pasa a ser parte de la secuencia de inputs a la hora de producir el próximo token, y así sucesivamente. Esta idea se conoce como *auto-regresión*. En contraste, BERT no posee esta característica, aunque ello significa que BERT puede incorporar el contexto izquierdo y derecho de cada palabra para obtener mejores resultados.

## 2. Un prototipo de aplicación de GPT-2 a una tarea creativa de Natural Language Generation (NLG)

La herramienta **AdGenerator** https://natural.do/ad-generator-demo (desarrollada por **Natural Tech** https://natural.do) genera anuncios automáticos y competitivos para el servicio Google Ads. AdGenerator inyecta publicidad junto a los resultados de las consultas que se realizan a través del buscador Google.

Cualquier anuncio de Google Ads tiene dos partes fundamentales que influyen en la atención del público: “slogan” y “descripción”. Nuestro sistema propone textos (*Natural Language Generation* o NLG) para completar cada una de dichas partes y producir de esta forma anuncios con el formato de Google Ads (**figura 6**).

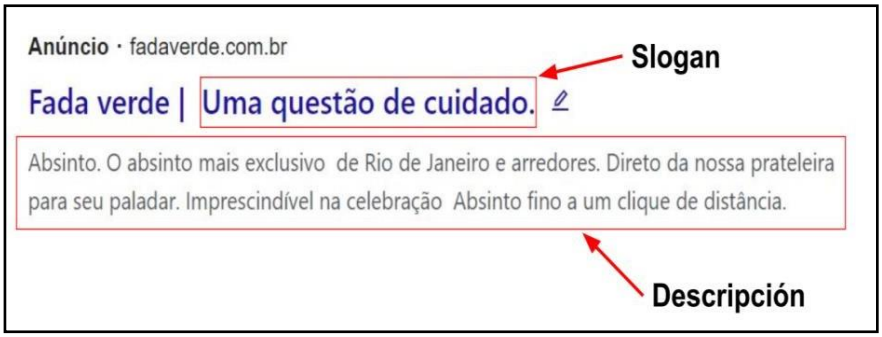

Figura 6: Formato de anuncio Google Ads

En esencia la herramienta presentada combina un algoritmo basado en GPT-2 entrenado para portugués brasileño, ajustado para publicidad, junto con las capacidades del algoritmo SentiLecto PT NLG también desarrollado por Natural Tech. El funcionamiento es simple. Se establece una serie de entradas (nombre de la empresa, características del producto, etc.) que alimentan los algoritmos y se generan textos relevantes que sirven como "slogan" y como "descripción" de un anuncio.

Concretamente, los modelos de GPT-2 utilizados se encargan de formar el slogan de un nuevo aviso publicitario mientras que SentiLecto desarrolla su descripción. Adicionalmente, la aplicación le brinda al usuario la posibilidad de hacer algunos ajustes manuales posteriores para alcanzar el resultado final.

### 2.1. GPT-2 como generador de slogans

El mercado de consumo brasileño se puede segmentar en grandes hiperdominios de productos, como alimentos y bebidas, vestuario y accesorios, etc. Por supuesto, cada uno de ellos comprende dominios más específicos. AdGenerator cuenta con un modelo GPT-2 optimizado para cada hiperdominio, es decir, se usó un conjunto de entrenamiento independiente por grupo (corpus de entrenamiento).

Los corpus de entrenamiento se componen de diferentes dominios de productos del mercado brasileño (cada corpus representa un

hiperdominio) y frases asociadas a ellos, en idioma portugués, recolectadas de Internet, que tienen una semántica adecuada para entrenar los modelos de forma que permitan generar nuevos textos creativos que llamen la atención.

## 2.2. Funcionamiento general de Ad Generator

Para construir un anuncio, en primer lugar Ad Generator solicita al usuario información de su empresa y del producto que se desea promocionar. Por ejemplo, un emprendimiento de venta de ajenjo tendría los siguientes datos (en portugués):

- Nombre de la empresa (obligatorio): *Fada Verde.*
- Descripción del producto (opcional): -
- Segmento o dominio del producto (obligatorio): *absinto.*
- Url de la empresa (opcional): *www.fadaverde.com.br*
- Ciudad de la empresa (opcional): *Rio de Janeiro.*

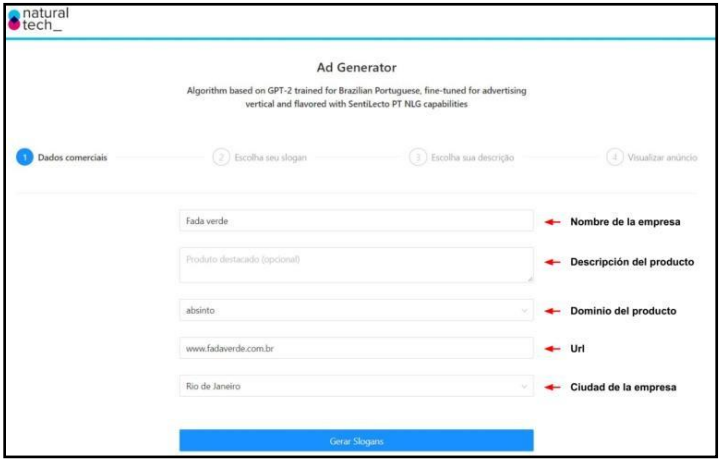


Figura 7: Datos solicitados por AdGenerator https://natural.do/ad-generator-demo

Al completar los campos requeridos (**figura 7**), se ejecuta el generador y la aplicación recomienda al usuario 20 frases que pueden servir de slogan para el anuncio.

En la **figura 8** se observan en los resultados frases como "*Uma questão de cuidado*" ("*Una cuestión de cuidado*") y "*experiência rara no paladar*" ("*experiencia rara en el paladar*"). Es importante entender que los modelos *transformers* deben "traducir" los textos con los que se alimentan a representaciones numéricas de palabras y dichas representaciones en algún grado están contextualizadas. Eso quiere decir que la generación de los slogans conlleva un proceso dentro del cual se relaciona el texto ingresado al modelo ("absinto licor y bebidas espirituosas") con la información (textos webs) que se utilizó para pre-entrenarlos y luego optimizados (fine-tuning). Por ejemplo, la frase "*Uma questão de cuidado*" se puede interpretar en el contexto de que el ajenjo es una bebida con un alto nivel de alcohol y que su consumo ha generado controversias en el pasado.

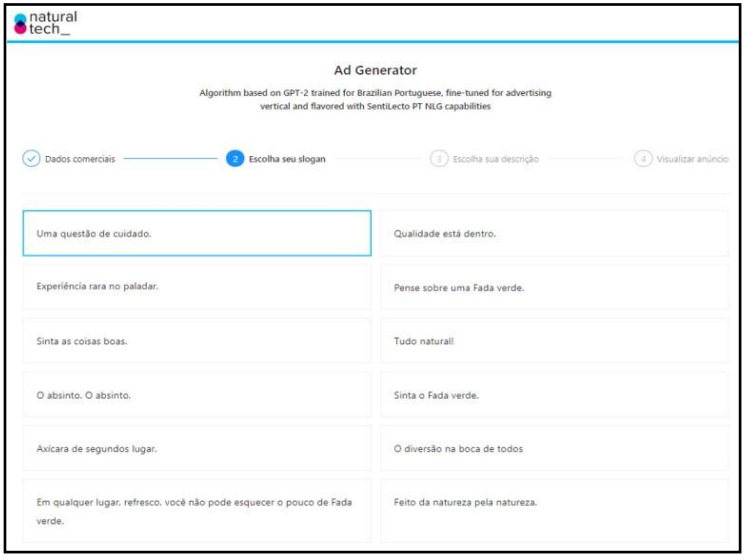


Figura 8: Slogans generados por la aplicación con GPT-2

Se selecciona "*Uma questão de cuidado*" y se sigue adelante. De esta forma, la herramienta conduce a otra ventana donde se origina nuevo texto para las descripciones del aviso. En esta ocasión el algoritmo SentiLecto PT NLG es el encargado de producir las oraciones de la descripción. Adicionalmente la tecnología establece palabras clave en el documento generado y brinda al usuario la capacidad de modificarlas por otras establecidas de acuerdo con el contexto del anuncio.

En el ejemplo (**figura 9**), se ha formado la frase "...*O absinto mais exclusivo de Rio de Janeiro...*" donde la palabra "*exclusivo*" se puede modificar por "*divertido*", "*saboroso*", etc.

Por último, se previsualiza el anuncio en la herramienta donde es posible hacer cambios manuales en la descripción y en el slogan.

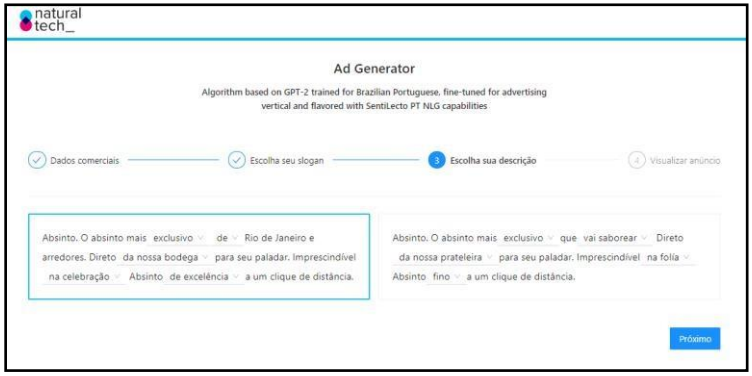

Figura 9: Descripciones generadas por la aplicación con SentiLecto PT NLG

## 2.3. *Fine-tuning* de GPT-2 como generador de slogans

GPT-2 es un modelo de lenguaje entrenado con 40 GB de texto de la web. Cuando se lo alimenta con una palabra, un fragmento de ella, o varias palabras (frase), tiene la capacidad de predecir la distribución de

probabilidad de la siguiente palabra considerando las palabras que le precedieron. Luego, se puede repetir la predicción una y otra vez hasta generar grandes espacios de texto coherente.

El dominio "absinto", usado en la demostración, concatenado con su primer dominio padre "licor & bebidas espirituosas", produce la frase "**absinto licor & bebidas espirituosas"** (**figura 10**).

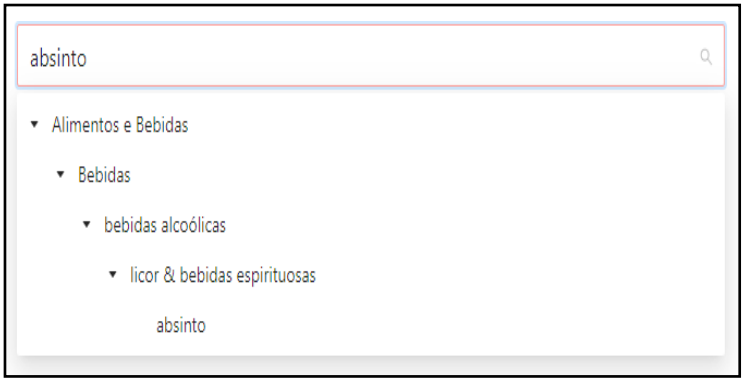


Figura 10: Dominios padres de "absinto"

Esta frase se *tokeniza* y se obtiene el siguiente resultado:

[absinto] [licor] [&] [bebidas] [espirituosas]

Los tokens usados representan el contexto que se le da al modelo a partir del cual se generan nuevas palabras, es decir, se predicen los siguientes tokens que formarán los slogans del anuncio.

**<contexto>** [absinto] [licor] [&] [bebidas] [espirituosas] **<slogan>** PREDICCIONES

En este caso, el modelo ajustado al hiperdominio "*Alimentos y Bebidas*" no tiene como entrada una descripción del producto que se desea anunciar. Debido a que el campo es opcional en la aplicación, el contexto se define por la descripción del producto (si la hubiera), el dominio de éste (campo obligatorio) y su categoría padre:

**<contexto>** [<dominio> <dominio padre> <descripción>] **<slogan>** PREDICCIONES

Se detalla que GPT-2 fue previamente entrenado con grandes longitudes de texto y originalmente no está diseñado para oraciones cortas como slogans. Por lo tanto, se rellenan las secuencias con tokens especiales para poder trabajar con una longitud variable de texto.

En la práctica, existen varios métodos para generar una secuencia de tokens a partir de un modelo de lenguaje. Algunos ejemplos son *greedy sampling*; *beam search*. *top-k sampling* y *top-p sampling*.

La técnica *greedy sampling* o *greedy search* consiste en elegir siempre el token con mayor probabilidad como la palabra siguiente. Esto conduce a resultados muy predecibles y repetitivos. El principal inconveniente de la búsqueda codiciosa es que omite palabras de alta probabilidad ocultas detrás de una palabra de baja probabilidad. En cambio, *beam search* genera múltiples secuencias al mismo tiempo y devuelve la secuencia cuya probabilidad general es la más alta. Esto significa que el algoritmo puede terminar eligiendo tokens con menor probabilidad en algún punto de la cadena de texto, pero conducirá a una secuencia final con mayor probabilidad que el procedimiento codicioso, aunque no se garantiza que encuentre la mejor salida, es decir, la más probable.

## Conclusión

A lo largo de este artículo, hemos cubierto sucintamente el resurgimiento y la vertiginosa evolución del paradigma conexionista bajo la oleada de los

modelos Deep Learning, ampliamente adoptados por la academia y por la industria en esta última década. Nos hemos adentrado en las principales características de las arquitecturas de las redes neuronales básicas y, luego, a partir de la revolución de los *Transformers*, hemos abarcado los modelos más recientes de código abierto para familiarizarnos con las grandes operaciones de pre-entrenamiento y *fine-tuning* de dichos modelos en tareas generales de PLN. Por último, hemos presentado en detalle la implementación de un prototipo basado en uno de los modelos más exitosos (GPT-2), refinado para una tarea específica de generación de texto comercial (slogans).

Debido a lo innovador de la tarea propuesta a modo de ejemplo de implementación en una lengua de escasos recursos de PLN y a las dificultades propias del campo NLG para dar con métricas de evaluación estándares, nuestro producto no ha sido aun rigurosamente evaluado. No obstante, confiamos en que pueda ejemplificar la cada vez más amplia adopción de estas arquitecturas de Deep Learning para tareas de entendimiento y generación de lenguaje natural en la industria del software y la Inteligencia Artificial.

Aportamos este artículo en la tradición de la divulgación científica tendiente al entendimiento de estos exitosos modelos de Deep Leaning, justamente en el momento coincidente con la cresta de la ola de adopción de estas tecnologías. No obstante, más allá del evidente interés científico-tecnológico, los recientes avances en el campo nos interpelan también desde un costado filosófico y hasta epistemológico. Cabe preguntarse si estos complejos algoritmos, que encuentran correlaciones infinitesimales entre palabras, están realmente aprendiendo a comprender nuestro lenguaje natural. Dejamos este interrogante pendiente como una fascinante puerta abierta a lo que deparen los más recientes avances en el campo, con un poder de cómputo que desafía la definición de inteligencia (artificial o humana).

**Referencias bibliográficas**

Bahdanau, D., Cho, K. y Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. En *ICLR 2015*. Recuperado de https://arxiv.org/abs/1409.0473.

Bastings, J., Titov, I., Aziz, W., Marcheggiani, D. y Sima'an, K. (2017). Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. En *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (pp. 1957-1967). Recuperado de https://www.aclweb.org/anthology/D17-1209.pdf

Bradbury, J., Merity, S., Xiong, C. y Socher, R. (2017). Quasi-Recurrent Neural Networks. En *ICLR 2017*. Recuperado de http://arxiv.org/abs/1611.01576.

Dai, A. M. y Le, Q. V. (2015). Semi-supervised Sequence Learning. En *Advances in Neural Information Processing Systems (NIPS '15)*, (pp. 1-9). Recuperado de https://papers.nips.cc/paper/5949-semi-supervised-sequence-learning.pdf.

Devlin, J., Chang, M., Lee K. y Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. En *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies, Volume 1 (Long and Short Papers)*, (pp. 4171-4186). Recuperado de https://www.aclweb.org/anthology/N19-1423.pdf.

Dyer, C., Kuncoro, A., Ballesteros, M. y Smith, N. A. (2016). Recurrent Neural Network Grammars. En *NAACL*. Recuperado de: http://arxiv.org/abs/1602.07776.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2), 179-211.

Gatt, A. y Krahmer, E. (2018). Survey of the State of the Art in Natural Language Generation: Core tasks, applications, and evaluation. *Journal of Artificial Intelligence Research*, 61, 65-170.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Gómez Colmenajero, S., Grefenstette, E., Ramalho, T., Agapiou, J., Puigdomènech Badia, A., Moritz Hermann, K., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K. y Hassabis, D. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature,* 538, 471-476.

Henaff, M., Weston, J., Szlam, A., Bordes, A. y LeCun, Y. (2017). Tracking the World State with Recurrent Entity Networks. En *Proceedings of ICLR 2017*.

Hochreiter, S. y Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.

Howard, J. y Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. En *Proceedings of ACL 2018*, (pp. 328-339). Recuperado de https://www.aclweb.org/anthology/P18-1031.pdf.

Kalchbrenner, N., Grefenstette, E. y Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. En *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, (pp. 655–665). Recuperado de http://arxiv.org/abs/1404.2188.

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A. van den, Graves, A. y Kavukcuoglu, K. (2016). Neural Machine Translation in Linear Time. ArXiv Preprint ArXiv: Recuperado de http://arxiv.org/abs/1610.10099.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. En *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, (pp. 1746–1751). Recuperado de http://arxiv.org/abs/1408.5882.

Kumar, A., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R. y Socher, R. (2016). Ask me anything: Dynamic memory networks for natural language processing. En *International Conference on Machine Learning*, (pp. 1378-1387). Recuperado de https://arxiv.org/pdf/1506.07285.pdf.

Levy, O. y Goldberg, Y. (2014). Dependency-Based Word Embeddings. En *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, (pp. 302–308). Recuperado de https://doi.org/10.3115/v1/P14-2050.

Merity, S. Shirish Keskar, N. y Socher, R. (2017). Regularizing and Optimizing LSTM Language Models. Recuperado de https://arxiv.org/pdf/1708.02182.pdf.

Mikolov, T., Sutskever, I., Chen, Kia., Corrado, G. y Dean, J. (2013). Distributed representations of words and phrases and their compositionality. En *Proceedings of NAACL-HLT 2018*, (pp. 2227–2237). Recuperado de https://aclweb.org/anthology/N18-1202.

Pennington, J., Socher, R. y Manning, C. (2014). Glove: Global vectors for word representation. En *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 1532–1543). Recuperado de https://www.aclweb.org/anthology/D14-1162.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. y Zettlemoyer, L. (2018). Deep contextualized word representations. En *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies, Volume 1 (Long Papers)*, (pp. 2227-2237). Recuperado de https://www.aclweb.org/anthology/N18-1202.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. y Sutskever, I. (2019). Language Models Are Unsupervised Multitask Learners [blog]. *OpenAI Blog*, 1, 8.

Ramachandran, P., Liu, P. J. y Le, Q. V. (2017). Unsupervised Pretraining for Sequence to Sequence Learning. En *Proceedings of EMNLP 2017*.

Ruder, S. (2018). A review of the recent history of NLP [blog]. Recuperado de https://ruder.io/a-review-of-the-recent-history-of-nlp/.

Socher, R., Perelygin, A. y Wu, J. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. En *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, (pp. 1631–1642).

Sutskever, I., Vinyals, O. y Le, Q. V. (2014). Sequence to sequence learning with neural networks. En *Advances in Neural Information Processing Systems (NIPS '14)*. Recuperado de: https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Sukhbaatar, S., Szlam, A., Weston, J. y Fergus, R. (2015). End-To-End Memory Networks. En *Proceedings of NIPS 2015*. Recuperado de http://arxiv.org/abs/1503.08895.

Subramanian, S., Trischler, A., Bengio, Y. y Pal, C. J. (2018). Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. En *Proceedings of ICLR 2018*.

Tai, K. S., Socher, R. y Manning, C. D. (2015). Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. En *ACL 2015*, (pp. 1556–1566).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. y Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems (NIPS)*, 1-11. Recuperado de https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Wang, J., Yu, L., Lai, K. R. y Zhang, X. (2016). Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model. En *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, (pp. 225–230).

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. y Bowman, S. (2019a). GLUE: A multi-task benchmark and analysis platform for natural language understanding. En *International Conference on Learning Representations*. Recuperado de https://openreview. net/forum?id=rJ4km2R5t7

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O. y Bowman, S. (2019b). SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. En *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*. Recuperado de https://w4ngatang.github.io/static/papers/superglue.pdf.

**Notas biográficas**

**Fernando Balbachan**

Fernando Balbachan es Doctor en Lingüística por la Universidad de Buenos Aires, Máster en Lingüística Computacional por la Indiana University y Licenciado en Letras con orientación en Lingüística por la Universidad de Buenos Aires (UBA). Se desempeña como Jefe de Trabajos Prácticos en la cátedra Modelos Formales No Transformacionales (UBA) y desde 2020 forma parte del cuerpo docente de la Especialización de Posgrado en Tecnologias aplicadas aos novos modelos transacionais de la Universidade Tuiuti do Paraná (Brasil). En el ámbito profesional se ha desempeñado como consultor de proyectos nacionales e internacionales en Procesamiento del Lenguaje Natural, fundando su propia compañía de NLP Natural Tech https://natural.do en 2015.

**Ignacio Maltagliatti**

Ignacio Maltagliatti es Ingeniero Civil por la Universidad Tecnológica Nacional (UTN, Argentina) y Máster en Data Mining (UTN, en curso). Se desempeña como desarrollador Senior de sistemas de NLP en la empresa Natural Tech desde 2020.

**Natalia Flechas**

Licenciada en Letras con orientación en Lingüística (UBA 2019). Adscripta a la cátedra de Modelos Formales No Transformacionales (UBA). Ha venido trabajando como lingüista computacional en diversas compañías de software.

**Francisco Pensa**

Experto programador y líder técnico proveniente del mundo de las ONGs y datos abiertos. Se ha venido desempeñando como CTO en la empresa Natural Tech desde 2020.

**Lucas Ramírez**

Licenciatura en Letras (UBA en curso). Consultor en capacitación y director de cursos de extensión universitaria en diversas universidades. Se ha venido desempeñando como Analista NLP Senior en la empresa Natural Tech desde 2020.