Implementación de modelos Transformers en conjuntos de datos reducidos

Autor 1
Escuela de ...
Uni...
..del Perú
Lima, Perú
...@uni.pe

Autor 2
Escuela de...
Universidad ...
.. del Perú
Lima, Perú
...@uni.pe

Autor 3
Escuela de Ciencia de la Computaci...
Univer...
... del Perú
Lima, Perú
...@uni.pe

***Resumen*—El modelado del lenguaje se ha abordado recientemente utilizando métodos de entrenamiento no supervisados como ELMo y BERT. Sin embargo, sigue siendo un desafío equipar adecuadamente las redes neuronales con una dependencia a largo plazo.**

En efecto, pese a que los modelos secuencia a secuencia son bastante versátiles y se utilizan en una variedad de tareas de NLP, una desventaja crítica es el diseño vectorial del contexto de longitud fija que no puede recordar oraciones largas. Un enfoque para resolver el problema de la pérdida de información relevante en oraciones largas es utilizar el mecanismo de atención que necesita calcular un valor para cada combinación de palabras de entrada y salida, por lo que suele ser costoso. En arquitecturas que utilizan el modelo Transformer, se utiliza variantes del mecanismo de atención para aprender también las dependencias de entrada y salida y pese al potencial de este algoritmo, tienen el problema de no extenderse más allá de un cierto nivel debido al uso del contexto de longitud fija no puede modelar dependencias que son más largas que una longitud fija.

Para resolver estos problemas se plantea un nuevo modelo llamado Transformer-XL, que forma parte de un nuevo conjunto de técnicas utilizadas para entrenar modelos altamente eficientes y de alto rendimiento en NLP.

En este trabajo se implementan sobre conjuntos reducidos como Penn Treebank varias de las técnicas recientes como el Transformer- XL, XLNet, que son una variante del Transformer.

Índice de Términos— RNN, modelos seq2seq, mecanismos de atención, transformers, Hugging Face.

I. Introducción

Una tendencia dada en el ICRL [1] y en el NAACL-HLT [2] celebrados en el 2019, muestra que las RNN tienen una mayor caída en publicaciones. Esto no es sorprendente porque, aunque los RNN son intuitivos para los datos secuenciales, sufren un inconveniente masivo: no pueden ser paralelizados y, por lo tanto, no pueden aprovechar el factor más importante que ha impulsado el progreso en la investigación desde 2012: la potencia de cálculo. Los RNN nunca han sido populares en visión computacional y aprendizaje por refuerzo y para NLP, están siendo reemplazados por arquitecturas basadas en la atención.

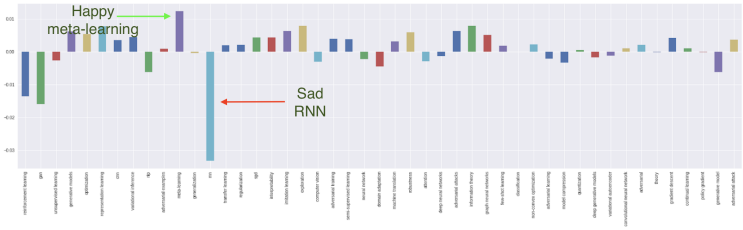

Figura 1. Estadisticas de ICRL 2019

II. Marco Teórico

II-A. El modelo secuencia a secuencias

El modelo secuencia a secuencia o seq2seq nació en el campo del modelamiento del lenguaje [6] y tiene como objetivo transformar una secuencia de entrada, en una nueva secuencias, ambas de longitudes arbitrarias. Ejemplos que utilizan estos modelos incluyen la traducción automática entre varios lenguajes de texto o audio, generación de diálogos de preguntas y respuestas, correctores ortográficos o incluso analizar oraciones en árboles de gramática.

El modelo seq2seq normalmente tiene una arquitectura encoder-decoder, compuesta de:

1. Un encoder procesa la secuencia de entrada y comprime la información en un vector de contexto, de longitud fija. Esta representación es resumen del significado de toda la secuencia de entrada.
2. Un decoder se inicializa con el vector de contexto para emitir una salida transformada.
   Tanto el encoder y decoder son redes neuronales recurrentes, es decir, que utilizan unidades LSTM [17], [18] o GRU [19].

En [20], se realiza una búsqueda entre diferentes arquitecturas RNN incluyendo el GRU en tareas como el

[1] International Conference on Learning Representations
[2] Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

modelamiento de lenguaje, recomendando agregar un sesgo de 1 a la puerta de olvido de cada LSTM en cada aplicación, lo que resultan en un mejor rendimiento.

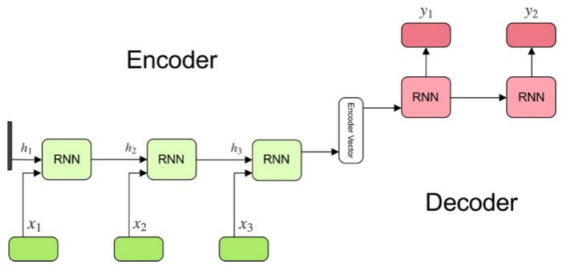

Figura 2. Estructura de un modelo seq2seq

II-A1. Problemas con los modelos seq2seq: Pese a que los modelos encoder-decoder son una subclase ampliamente utilizada tiene ciertos incovenientes:

- El cálculo secuencial evita la paralelización.
- Comprime toda la información de una oración de entrada en un vector de longitud fija que luego toma el decoder, lo que conduce a una disminución en el rendimiento cuando se trata de oraciones largas.

El mecanismo de atención nació propuesto por Bahdanau [2] apareció para resolver algunos de estos problemas.

II-B. Mecanismos de atención

En estos modelos, cada vez que el modelo predice una palabra de salida, solo usa partes de una entrada donde se concentra la información más relevante en lugar de una oración completa. El encoder funciona como de costumbre, pero el estado oculto del decoder se calcula con un vector de contexto, la salida anterior y el estado oculto anterior. Los vectores de contexto se calculan como una suma ponderada de anotaciones generadas por el encoder con un vector de contexto separado para cada palabra objetivo.

Sean $\mathbf{x}$ una secuencia de entrada de longitud de n y sea $\mathbf{y}$ una secuencia de longitud m a generar:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]$$
$$\mathbf{y} = [y_1, y_2, \ldots, y_m]$$

En el caso de un LSTM bidireccional, estas anotaciones son concatenaciones de estados ocultos en las direcciones hacia adelante $\overrightarrow{\boldsymbol{h}}_i$ y hacia atrás $\overleftarrow{\boldsymbol{h}}_i$. Una concatenación de dos representa el estado del encoder. La motivación es incluir tanto las palabras anteriores como las siguientes en la anotación de una palabra.

$$\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i^\top; \overleftarrow{\boldsymbol{h}}_i^\top]^\top, i = 1, \ldots, n$$

La red del decodificador tiene un estado oculto $\boldsymbol{s}_t = f(\boldsymbol{s}_{t-1}, y_{t-1}, \mathbf{c}_t)$, para la palabra de salida en la posición $t = 1, \ldots, m$, donde el vector de contexto $\mathbf{c}_t$, es una suma de estados ocultos de la secuencia de entrada, ponderada por puntajes de alineación:

$$\begin{aligned}\mathbf{c}_t &= \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i \\ \alpha_{t,i} &= \text{align}(y_t, x_i) \\ &= \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))}.\end{aligned}$$

El modelo de alineación asigna una puntuación $\alpha_{t,i}$ al par de entrada en la posición i y salida en la posición t, (y_t, x_i), en función de qué tan bien coinciden. El conjunto de $\{\alpha_{t,i}\}$ son pesos que definen cuánto de cada estado oculto de la fuente debe considerarse para cada salida.

En el artículo de Bahdanau, el puntaje de alineación α está parametrizado por una red con una sola capa oculta y esta red se entrena conjuntamente con otras partes del modelo. Por lo tanto, la función de puntuación tiene la siguiente forma, dado que tanh se usa como una función de activación no lineal:

$$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$$

donde ambos $\mathbf{v}_a$ y $\mathbf{W}_a$ son matrices de peso que se aprenderán en el modelo de alineación.

La matriz de puntajes de alineación muestra explícitamente la correlación entre las palabras de entrada y de salida.

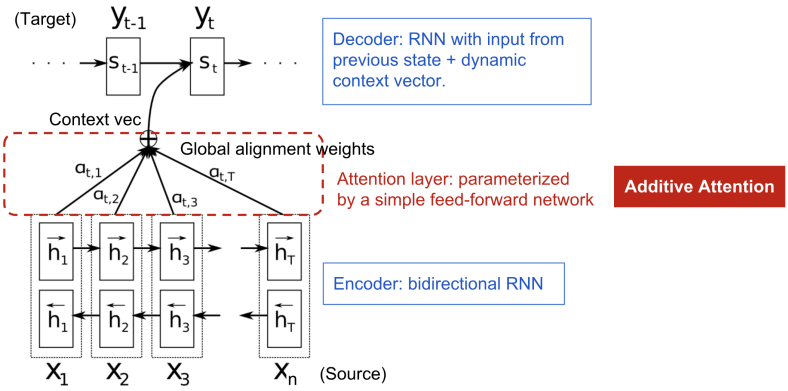

Figura 3. Modelo encoder-decoder con atención aditiva

II-C. Familia de mecanismo de atención

Dada la gran mejora de la atención en la traducción automática, pronto se extendió al campo de la visión computacional [5] y la gente comenzó a explorar otras formas de mecanismos de atención [1], [4], [7].

A continuación se muestra una tabla resumen de varios mecanismos de atención populares y las correspondientes funciones de puntuación de alineación:

Nombre	Alignment score función	Paper
Content-base attention	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \text{cosine}[\mathbf{s}_t, \mathbf{h}_i]$	**Graves 2014** [3]
Additive(*)	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{v}_a^T \tanh[\mathbf{s}_t, \mathbf{h}_i]$	**Bahdanau 2015** [2]
Location(**)	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \mathbf{s}_t)$	**Luong 2015** [4]
General(***)	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{s}_t^T \mathbf{W}_a \mathbf{h}_i$	**Luong 2015** [4]
Dot-Product	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{s}_t^T \mathbf{h}_i$	**Luong 2015** [4]
Scaled Dot-Product (****)	$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \frac{\mathbf{s}_t^T \mathbf{h}_i}{\sqrt{n}}$	**Waswani 2017** [1]

- (*) Conocido como `concat` en Luong [4] y como `additive attention` en Vaswani [1].
- (**) La ecuación simplifica la alineación softmax para que solo dependa de la posición del objetivo.
- (***) $\mathbf{W}_a$ es una matriz de pesos entrenables en la capa de atención.
- (****) Se agrega un factor de escala $1/\sqrt{n}$, motivada por la preocupación cuando la entrada es grande, el softmax puede tener un gradiente extremadamente pequeño, difícil para un aprendizaje eficiente.

Aquí hay un resumen de categorías más amplias de mecanismos de atención:

Nombre	Definición	Papers
Global	Utiliza todos los estados ocultos del encoder para calcular el vector de contexto.	**Xu 2015** [7]
Local	Elige una posición en la oración de entrada para determinar la ventana de palabras a considerar.	**Luong 2016** [6]
Auto-atencion	Relaciona diferentes posiciones de una sola secuencia para calcular su representación interna	**Cheng 2017** [13]
Bidireccional	El modelo atiende la hipótesis y la premisa y ambas representaciones se concatenan	**Seo 2017** [14]
Clave-Valor	Los vectores de salida se separan en claves para calcular la atención y los valores para codificar la distribución de la siguiente palabra y la representacion del contexto.	**Mino 2017** [15]
Jerarquía Anidada	Dos niveles de atención: primero a nivel de palabra y segundo a nivel de oración.	**Yang 2016** [16]

II-C1. Auto-atención: La auto-atención es un mecanismo de atención que relaciona diferentes posiciones de una sola secuencia para calcular una representación de la misma secuencia.

II-D. Transformers

El modelo Transformer, propuesto en el paper Attention Is All You Need, se basa en la auto-atención sin el uso de RNN. Como resultado, es altamente paralelo y requiere menos tiempo para entrenar, al tiempo que establece resultados de vanguardia en modelamiento de lenguaje y la traducción automática.

El Transformer se basa en una estructura encoder-decoder. La diferencia entre este y cualquier otro modelo es que el Transformer se basa completamente en mecanismos de atención y capas puntuales totalmente conectadas tanto para el encoder como para el decoder.

El componente principal en el Transformer es la unidad de mecanismo de auto-atención de múltiples cabeceras. El transformador ve la representación codificada de la entrada como un conjunto de pares clave-valor (K, V), ambos de dimensión n (longitud de secuencia de entrada). En el contexto de NMT, tanto las claves como los valores son estados ocultos del encoder. En el decoder, la salida anterior se comprime en una consulta (Q de dimensión m) y la siguiente salida se produce al mapear esta consulta y el conjunto de claves y valores.

El transformador adopta la atención escalada del producto punto: la salida es una suma ponderada de los valores, donde el peso asignado a cada valor está determinado por el producto punto de la consulta con todas las claves:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$

En lugar de solo calcular la atención una vez, el mecanismo de múltiples cabeceras corre la atención escalada del producto puntos varias veces en paralelo. Las salidas de atención independientes simplemente se concatenan y se transforman linealmente en las dimensiones esperadas.

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1; \ldots; \text{head}_h]\mathbf{W}^O$$
$$\text{donde head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

donde $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ y $\mathbf{W}^O$ son matrices de parámetros para aprender.

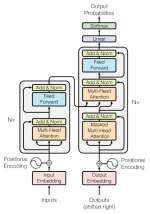

Figura 4. Arquitectura del Transformer

II-D1. Limitaciones del transformer: El transformer es una mejora con respecto a los modelos seq2seq basados en RNN. Pero tiene algunas limitaciones:

- La atención solo puede ocuparse de cadenas de texto de longitud fija. El texto debe dividirse en un cierto número de segmentos antes de ser alimentado al sistema como entrada.
- Esta segmentación del texto provoca la fragmentación del contexto [3]. Por ejemplo, si una oración se divide en dos, se pierde una cantidad significativa de contexto. En otras palabras, el texto se divide sin considerar la oración o cualquier otro límite semántico.

[3] context fragmentation

III. Estado del Arte

La arquitectura del transformador ya ha dado un paso adelante en 2019 con la publicación del transformer-xl [25] y otras arquitecturas relacionadas [24]–[29].

III-A. Transformador Universal

El Transformador Universal es un modelo de secuencia recurrente auto-atento en paralelo en el tiempo, que es paralelizable sobre la secuencia de entrada. Al igual que el transformador base, tiene un `campo receptivo global` (lo que significa que mira muchas palabras a la vez). La idea nueva y principal es que en cada paso recurrente, el transformador universal refina iterativamente sus representaciones para todos los símbolos de la secuencia utilizando la auto-atención seguida de una función de transición compartida en todas las posiciones y pasos de tiempo.

El Universal Transformer, combina el modelo original Transformer con una técnica llamada Adaptive Computation Time, un mecanismo que permite la aplicación del encoder un número variable de veces y la aplicación del decoder un número variable de veces.

El objetivo de Universal Transformer es lograr un buen rendimiento tanto en traducción de lenguaje como en tareas algorítmicas con un solo modelo. Los autores del Universal Transformer señalan que es un modelo completo de Turing.

III-B. BERT

BERT fue uno de los primeros modelos que muestra que los transformadores pueden alcanzar el rendimiento a nivel humano en una variedad de tareas basadas en el lenguaje: responder preguntas, clasificar sentimientos o clasificar si dos oraciones se suceden naturalmente.

BERT consiste en una simple pila de bloques transformadores. Esta pila está pre-entrenada en un gran corpus de dominio general que consta de 800 millones de palabras de libros en inglés y 2,5 mil palabras de texto de artículos de Wikipedia en inglés sin marcado y utiliza la tokenización WordPiece, que se encuentra entre las secuencias de nivel de palabra y nivel caracter.

BERT utiliza la tokenización de WordPiece, que se encuentra entre las secuencias de nivel de palabra y de nivel de palabra. Este divide palabras como walking en los tokens walk y ## ing. Esto permite que el modelo haga algunas inferencias basadas en la estructura de las palabras: dos verbos que terminan en -ing tienen funciones gramaticales similares, y dos verbos que comienzan con walk tienen una función semántica similar.

En un experimento de ablación, los autores muestran que la mayor mejora en comparación con los modelos anteriores proviene de la naturaleza bidireccional del BERT. Es decir, modelos anteriores como GPT usaban una máscara autoregresiva, que solo permitía atención sobre los tokens anteriores. El hecho de que en BERT toda la atención se centre en toda la secuencia es la causa principal del rendimiento mejorado. Es por eso que la **B** en BERT significa bidireccional.

El modelo BERT más grande utiliza 24 bloques de transformadores, una dimensión de integración de 1024 y 16 cabeceras de atención, lo que da como resultado parámetros de $340M$.

III-C. GPT-2

GPT2 es fundamentalmente un modelo de generación de lenguaje, utiliza la autoatención enmascarada. Utiliza BPE para simular el lenguaje, que, al igual que encoding WordPiece, divide las palabras en tokens que son un poco más grandes que caracteres individuales pero menos que palabras completas.

Los autores de GPT-2 crearon un nuevo conjunto de datos de alta calidad, para ello utilizaron sitios de redes sociales como Reddit para encontrar una gran colección de escritos con un cierto nivel mínimo de apoyo social (expresado en Reddit como karma).

GPT2 está construido de manera muy similar al modelo de generación de texto anterior, con solo pequeñas diferencias en el orden de las capas y trucos adicionales para entrenar a mayores profundidades. El modelo más grande usa 48 bloques transformadores, una longitud de secuencia de 1024 y una dimensión de embeddings de 1600, lo que da como resultado parámetros de $1{,}5B$.

Un apunte adicional es que el GPT-2 es el primer modelo transformador que se convirtió en noticia, después de la controvertida decisión del OpenAI de no lanzar el modelo completo. La razón fue que GPT-2 podría generar un texto lo suficientemente creíble como para que las campañas de noticias falsas a gran escala visto en las elecciones presidenciales de EE.UU. del 2016 se conviertan efectivamente en un trabajo de una sola persona.

III-D. Transformer-XL

Si bien el transformador representa un salto masivo positivo en el modelado de la dependencia de largo alcance, los modelos que hemos visto hasta ahora todavía están fundamentalmente limitados por el tamaño de la entrada. Dado que el tamaño de la matriz de producto punto crece cuadráticamente en la longitud de la secuencia, esto rápidamente se convierte en un cuello de botella a medida que intentamos extender la longitud de la secuencia de entrada. El transformer-XL es uno de los primeros modelos exitosos en abordar este problema.

En el paper [30] se propone este método novedoso para el modelado de lenguaje, que permite que una arquitectura transformer aprenda dependencia a largo plazo, a través de un mecanismo de recurrencia, más allá de una longitud fija sin alterar la coherencia temporal.

El método es diferente de otros enfoques anteriores que se centran en otras estrategias para soportar la dependencia a largo plazo, como las señales de pérdida adicionales [4] y la estructura de memoria aumentada [5].

Se introduce un mecanismo recurrente a nivel de segmento que permite que el modelo reutilice estados ocultos anteriores en el momento del entrenamiento, abordando tanto los problemas del contexto de longitud fija como la fragmentación del contexto. En otras palabras, la información histórica se puede reutilizar y se puede extender tanto como lo permita la memoria de la GPU.

Para reutilizar adecuadamente los estados ocultos, los autores proponen un mecanismo llamado codificaciones posicionales relativos que ayuda a evitar la confusión temporal. Los modelos actuales no pueden distinguir la diferencia posicional entre entradas en diferentes segmentos en diferentes capas. La codificación de posición relativa soluciona este problema al codificar el sesgo de información posicional en los estados ocultos, que difiere de otros enfoques que realizan esto como el nivel de entrada.

Como se trata de una arquitectura transformer, el proceso anterior se logra calculando la distancia relativa entre cada vector clave y el vector de consulta y colocando el resultado en el puntaje de atención. Con un nuevo truco de parametrización de los términos utilizados para derivar el puntaje de atención entre consulta y vector, se puede incorporar la información de posición relativa. El componente de recurrencia ahora está equipado con un embedding posicional relativa propuesto y todo este procedimiento representa la arquitectura transformer-XL.

III-D1. Resultados:

- Tanto para el modelado de lenguaje a nivel de palabra como a nivel de caracteres aplicado a una variedad de conjuntos de datos como WikiText-103, text8 y One Billion Word se obtiene resultados sólidos.
- El transformer-XL reduce el puntaje de perplejidad SoTA anterior en varios conjuntos de datos como text8, enwiki8, One Billion Word y WikiText-103. Los autores afirman que el método es más flexible, más rápido durante la evaluación (aceleración de 1874 veces), se generaliza bien en conjuntos de datos pequeños y es eficaz para modelar secuencias cortas y largas.
- Se proporciona un estudio de ablación para examinar los efectos tanto del mecanismo de recurrencia como del esquema de codificación posicional propuesto.
- Los autores proponen una nueva métrica llamada Relative Effective Context Length a que proporciona una manera justa de comparar modelos que se prueban con mayores longitudes de contexto.

[4] additional loss signals
[5] augmented memory structure

III-E. Transformadores dispersos

Los transformadores dispersos abordan el problema del uso de memoria cuadrática usadas en la cabeceras. En lugar de calcular una matriz densa de pesos de atención (que crece cuadráticamente), calculan la autoatención solo para pares particulares de tokens de entrada, lo que resulta en una matriz de atención dispersa, con solo $n\sqrt{n}$ elementos explícitos.

Esto permite modelos con tamaños de contexto muy grandes, por ejemplo para modelado generativo sobre imágenes, con grandes dependencias entre píxeles. La desventaja es que la estructura de dispersión no se aprende, por lo que al elegir una matriz dispersa, estamos inhabilitando algunas interacciones entre los tokens de entrada que podrían haber sido útiles. Sin embargo, dos unidades que no están directamente relacionadas aún pueden interactuar en las capas más altas del transformador (similar a si una red convolucional crea un campo receptivo más grande con más capas convolucionales).

Más allá del beneficio simple de los transformadores de entrenamiento con longitudes de secuencia muy grandes, el transformador disperso también permite una forma muy elegante de diseñar un sesgo inductivo. En esta arquitectura se toma una entrada como una colección de unidades (palabras, caracteres, píxeles en una imagen, nodos en un grafo) y especificamos a través de la dispersión de la matriz de atención, qué unidades creemos que están relacionadas. El resto es solo una cuestión de construir el transformador tan profundo como sea posible y ver si se entrena.

III-F. XLNet

XLNet es un modelo de lenguaje autoregresivo que genera la probabilidad conjunta de una secuencia de tokens basada en la arquitectura del transformador con recurrencia. Este modelo introduce una variante de modelado de lenguaje llamada modelado de lenguaje de permutación. Los modelos de lenguaje de permutación están entrenados para predecir un token dado el contexto anterior como el modelo de lenguaje tradicional, pero en lugar de predecir los tokens en orden secuencial, predice los tokens en un orden aleatorio.

Además de usar el modelado de lenguaje de permutación, XLNet mejora BERT al usar el Transformer XL como su arquitectura base. El Transformer XL mostró un rendimiento de vanguardia en el modelado de lenguajes, por lo que fue una elección natural para XLNet.

XLNet utiliza las dos ideas clave de Transformer XL: embeddings posicionales relativas y el mecanismo de recurrencia. Los estados ocultos del segmento anterior se almacenan en caché y se congelan mientras se realiza el

modelado del lenguaje de permutación para el segmento actual. Como todas las palabras del segmento anterior se usan como entrada, no es necesario conocer el orden de permutación del segmento anterior.

Los autores descubrieron que el uso del Transformer XL mejoraba el rendimiento sobre BERT, incluso en ausencia de modelado de lenguaje de permutación. Esto muestra que mejores modelos de lenguaje pueden conducir a mejores representaciones y, por lo tanto, a un mejor rendimiento en una multitud de tareas, lo que motiva la necesidad de investigar el modelado del lenguaje.

IV. Metodología

V. Experimentación y resultados

VI. Conclusiones

VII. Discusión

VIII. Conclusiones y trabajos futuros

Referencias

[1] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner,Christopher Clark, Kenton Lee, Luke Zettlemoyer. Deep contextualized word representations. NAACL 2018.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018). Disponible en https://arxiv.org/abs/1810.04805.

[3] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pp. 5998-6008. 2017. Disponible en https://arxiv.org/abs/1706.03762.

[4] Bahdanau, D., Cho, K., Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. Disponible en https://arxiv.org/abs/1409.0473.

[5] Alex Graves, Greg Wayne, Ivo Danihelka. Neural Turing Machines (2014). Disponible en https://arxiv.org/abs/1410.5401.

[6] Luong, Minh-Thang, Hieu Pham y Christopher D. Manning. Effective approaches to attention-based neural machine translation (2015). Disponible en https://arxiv.org/abs/1508.04025.

[7] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

[8] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. Advances in neural information processing systems. 2014. Disponible en https://arxiv.org/abs/1409.3215

[9] Denny Britz, Anna Goldie, Thang Luong y Quoc Le. Massive exploration of neural machine translation architectures. ACL 2017. Disponible en https://arxiv.org/abs/1703.03906.

[10] Kishore Papineni , Salim Roukos , Todd Ward , Wei-jing Zhu (2002). BLEU: a Method for Automatic Evaluation of Machine Translation, pp. 311-318. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL).

[11] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, Koray Kavukcuoglu. Neural Machine Translation in Linear Time (2017). Disponible en https://arxiv.org/abs/1610.10099. Hugging Face

[12] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin. Convolutional Sequence to Sequence Learning (2017). Disponible en https://arxiv.org/abs/1705.03122.

[13] Jianpeng Cheng, Li Dong y Mirella Lapata. Long short-term memory-networks for machine reading. EMNLP 2016. Disponible en https://arxiv.org/abs/1601.06733.

[14] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hannaneh Hajishirzi. Bidirectional Attention Flow for Machine Comprehension. Published as a conference paper at ICLR 2017. Disponible en https://arxiv.org/abs/1611.01603.

[15] Hideya Mino, Masao Utiyama, Eiichiro Sumita, Takenobu Tokunaga.Key-value Attention Mechanism for Neural Machine Translation.Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2017. Disponible en https://www.aclweb.org/anthology/I17-2049/.

[16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, Eduard Hovy. Hierarchical Attention Networks for Document Classification. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (2016). Disponible en https://www.aclweb.org/anthology/N16-1174/.

[17] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization (2014) Disponible en https://arxiv.org/abs/1412.6980.

[18] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1): 1929-1958, 2014. Disponible en http://jmlr.org/papers/v15/srivastava14a.html.

[19] Sergey Ioffe, Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (2015). Disponible en https://arxiv.org/abs/1502.03167.

[20] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E.Hinton. Layer Normalization (2016). Disponible en https://arxiv.org/abs/1607.06450.

[21] Sergey Zagoruyko, Nikos Komodakis. Wide Residual Networks (2017). Disponible en https://arxiv.org/abs/1605.07146.

[22] Sepp Hochreiter, Jürgen Schmidhuber. Long short-term memory.Neural computation,9(8):1735-1780, 1997.

[23] Understanding LSTM Networks, Olah, C., 2015. Enlace: http://colah.github.io/posts/2015-08-Understanding-LSTMs.

[24] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation (2014). Disponible en https://arxiv.org/abs/1406.1078.

[25] Rafal Jozefowicz, Wojciech Zaremba, Ilya Sutskever. An empirical exploration of recurrent network architectures. ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, Pages 2342-2350 (2015).

[26] Alex Graves. Adaptive Computation Time for Recurrent Neural Networks (2017). Disponible en https://arxiv.org/abs/1603.08983.

[27] Łukasz Kaiser, Ilya Sutskever. Neural GPUs Learn Algorithms, 2015. CoRR, Vol abs/1511.08228. Disponible en https://arxiv.org/abs/1511.08228.

[28] Oriol Vinyals, Meire Fortunato, Navdeep Jaitly. Pointer Networks, 2015. Advances in Neural Information Processing Systems, pp. 2692-2700. Disponible en https://arxiv.org/abs/1506.03134.

[29] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, Łukasz Kaiser. Universal Transformers (2018). ICLR 2019. Disponible en https://arxiv.org/abs/1807.03819.

[30] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context (2019). Disponible en https://arxiv.org/abs/1901.02860.

[31] M. Schuster and K. Nakajima. Japanese and Korea Voice Search (2012). Disponible https://storage.googleapis.com/pub-tools-public-publication-data/pdf/37842.pdf.

[32] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding (2019). Disponible en https://arxiv.org/abs/1906.08237.

[33] Pytorch-Transform:Implementación de transformer para NLP. https://pytorch.org/hub/huggingface_pytorch-transformers/.

[34] Martin Popel, Ondrej Bojar.Training Tips for the Transformer Model (2018). Disponible en https://arxiv.org/abs/1804.00247.

[35] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Jamie Brew. Transformers: State-of-the-art Natural Language Processing (2019), HuggingFace Inc. Disponible en https://arxiv.org/abs/1910.03771.